Planar Segmentation of Range Images

by

Simon Adriaan Muller

Thesis presented in partial fulfilment of the requirements for the degree of
Master of Science in Applied Mathematics
at Stellenbosch University

Supervisors: Dr W.H. Brink & Prof. B.M. Herbst

Department of Mathematical Sciences
Faculty of Science

March 2013

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Simon Muller
Date: March 2013

Abstract

Range images are images that store at each pixel the distance between the sensor and a particular point in the observed scene, instead of the colour information. They provide a convenient storage format for 3-D point cloud information captured from a single point of view. Range image segmentation is the process of grouping the pixels of a range image into regions of points that belong to the same surface. Segmentations are useful for many applications that require higher-level information, and with range images they also represent a significant step towards complete scene reconstruction.

This study considers the segmentation of range images into planar surfaces. It discusses the theory and also implements and evaluates some current approaches found in the literature. The study then develops a new approach based on the theory of graph cut optimization which has been successfully applied to various other image processing tasks but, according to a search of the literature, has otherwise not been used to attempt segmenting range images.

This new approach is notable for its strong guarantees in optimizing a specific energy function which has a rigorous theoretical underpinning for handling noise in images. It proves to be very robust to noise and also different values of the few parameters that need to be trained. Results are evaluated in a quantitative manner using a standard evaluation framework and datasets that allow us to compare against various other approaches found in the literature. We find that our approach delivers results that are competitive when compared to the current state-of-the-art, and can easily be applied to images captured with different techniques that present varying noise and processing challenges.

Opsomming

Dieptebeelde is beelde wat vir elke piksel die afstand tussen die sensor en 'n spesifieke punt in die waargenome toneel, in plaas van die kleur, stoor. Dit verskaf 'n gerieflike stoorformaat vir 3-D puntwolke wat vanaf 'n enkele sigpunt opgeneem is. Die segmentasie van dieptebeelde is die proses waarby die piksels van 'n dieptebeeld in gebiede opgedeel word, sodat punte saam gegroepeer word as hulle op dieselfde oppervlak lê. Segmentasie is nuttig vir verskeie toepassings wat hoërvlak inligting benodig en, in die geval van dieptebeelde, verteenwoordig dit 'n beduidende stap in die rigting van volledige toneel-rekonstruksie.

Hierdie studie ondersoek segmentasie waar dieptebeelde opgedeel word in plat vlakke. Dit bespreek die teorie, en implementeer en evalueer ook sekere van die huidige tegnieke wat in die literatuur gevind kan word. Die studie ontwikkel dan 'n nuwe tegniek wat gebaseer is op die teorie van grafieksnit-optimering wat al suksesvol toegepas is op verskeie ander beeldverwerkingsprobleme maar, sover 'n studie op die literatuur wys, nog nie gebruik is om dieptebeelde te segmenteer nie.

Hierdie nuwe benadering is merkbaar vir sy sterk waarborge vir die optimering van 'n spesifieke energie-funksie wat 'n sterk teoretiese fondasie het vir die hantering van geraas in beelde. Die tegniek bewys om fors te wees tot geraas sowel as die keuse van waardes vir die min parameters wat afgerig moet word. Resultate word geëvalueer op 'n kwantitatiewe wyse deur die gebruik van 'n standaard evalueringsraamwerk en datastelle wat ons toelaat om hierdie tegniek te vergelyk met ander tegnieke in die literatuur. Ons vind dat ons tegniek resultate lewer wat mededingend is ten opsigte van die huidige stand-van-die-kuns en dat ons dit maklik kan toepas op beelde wat deur verskeie tegnieke opgeneem is, alhoewel hulle verskillende geraastipes en verwerkingsuitdagings bied.

Contents

Chapter 1

Introduction

The field of computer vision encompasses methods for acquiring, processing, analysing and understanding images. The different applications of computer vision can often be categorized into:

- **restoring** the original image before corruption by noise;
- **reconstructing** the underlying model of a scene from one or more images;
- **recognizing** whether a certain event or object occurs.

This study falls into the category of scene reconstruction whose ultimate goal is to have a greater understanding of an image by modelling the underlying structure that caused it. Having this structural model is useful for many applications, some of which are discussed in the following section.

1.1 Motivation

Scene reconstruction can be useful in the decision making processes for automated tasks that employ vision, such as navigational planning, object recognition and object manipulation. While raw image data varies due to intricate combinations of multiple environmental factors, an ideal reconstruction of a scene contains only the actual physical structures that are present. While an ideal scene reconstruction should be more robust to environmental conditions affecting the captured data, it is still highly dependent on the robustness of the methods used to obtain the reconstruction. Furthermore, while the raw data that is captured is usually very dense in information, the underlying structures of the scene are often simpler and require much less information to be represented. Less information means that less memory and computational power would be required to process the data.

A significant advantage of a scene reconstruction is that it is a higher-level description that allows more advanced inference to be made about unseen aspects of the scene. For instance, a surface reconstruction which identifies that a large collection of points are actually three adjacent surfaces might, depending on the orientation and positioning of the surfaces, enable the system to infer which larger object they represent. This allows a type of object recognition where an object is identified by a combination of its parts. When only one viewpoint of a scene is available this type of object identification, or even just some simple assumptions about the full 3-D structure of a partially seen object, allows more intelligent choices to be made. For instance, a vehicle trying to find the optimal path to navigate a cluttered terrain can make assumptions about traversable pathways that are not directly visible because they lie behind other objects. This same ability would help a robotic arm with object manipulation tasks as it can make intelligent assumptions about the overall shape of an object by observing it from only one viewpoint.

1.2 Problem statement

Within the larger goal of achieving a structural model of a scene, this study is limited to investigating options in the field of range image segmentation, with a focus on planar segmentation. The motivations for using range images and studying planar segmentation are discussed in the following subsections.

1.2.1 Use of range images

The first goal in achieving a scene reconstruction is often to obtain the distances from the camera to various points in the image. While obtaining *relative* distances between points or objects is possible with colour images by using a variety of visual cues (as in human monocular vision), this study only consider data and methods where *absolute* distances to points are available.

As long as the data is captured in such a manner as to make it possible to obtain absolute distances to all points, then a point cloud can be constructed, which represents the underlying scene model while being independent of the capturing method*. This study assumes that a range image can be obtained from the captured data, along with a scaling algorithm that allows absolute distances to be calculated and a point cloud to be generated from the range image.

An advantage of using range data as opposed to colour images is that there is no longer ambiguity inherent in analysing colour or intensity data where the colour of each point is determined by an intricate combination of unknown environmental factors like lighting conditions, texture and reflection properties of surfaces, structure, viewing angle, etc. A range image maintains only the information that is of interest in understanding the 3-dimensional structure of the scene. This comes at the cost of information that can be inferred from colour images, like the properties of the materials in the scene and the sources of lighting and shadow. How range images are obtained (in some cases from colour images), and also a more detailed definition of what they are, is discussed in the following section.

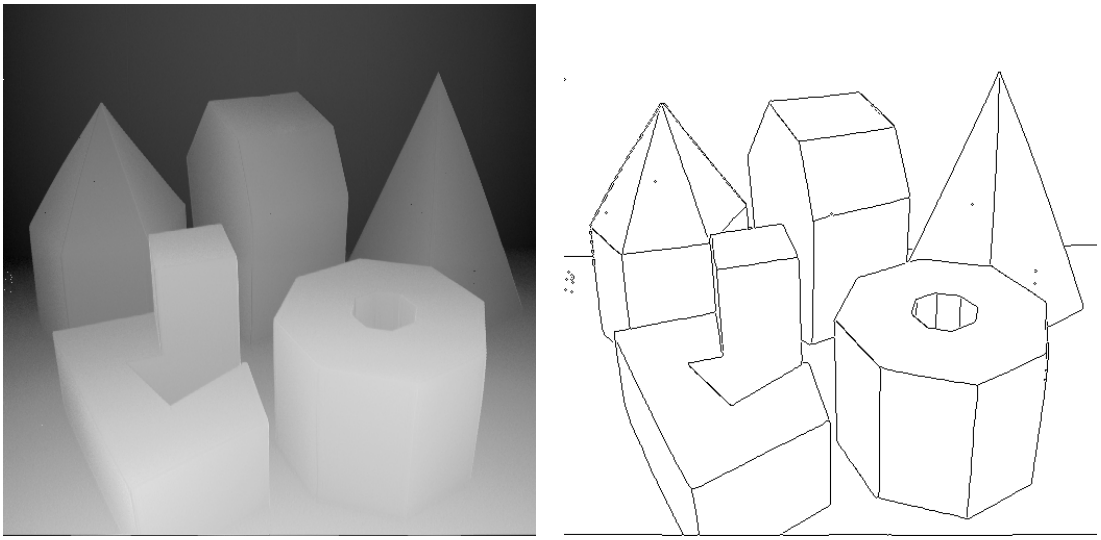
1.2.2 Definition of a range image

An image is usually represented as a 2-dimensional array of pixel values. Most commonly, these values represent either the intensity of light, as in a grey-scale image, or the colour, as in any colour image. In the case of range images, these values represent distances from the sensor to particular points in the scene. An example is shown in figure 1.1. Each method of generating a range image (discussed below) also requires its own scaling algorithm and sensor calibration parameters so that the pixels of the range image can be scaled to their actual positions as measured in metric coordinates from the sensor.

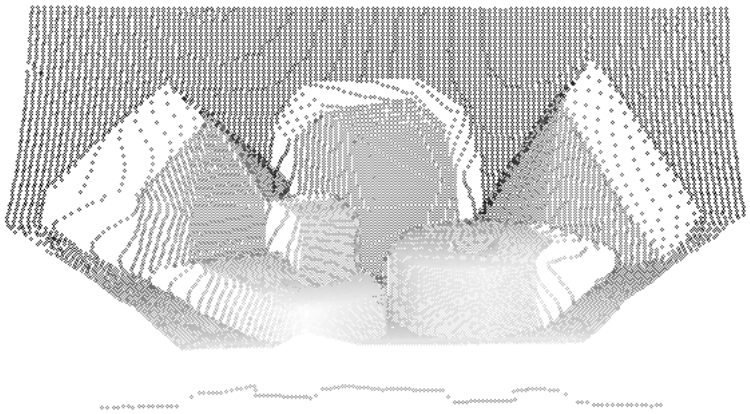
This study will assume that range images conform to the following requirements.

- Pixels are organized into a rectangular grid structure (a raster image) which implies that there exists a neighbourhood structure where pixels can have no more than eight direct neighbours (top, down, left, right and four corners).
- All pixel values in the image were obtained from a single vantage point.
- There is a limited range of possible values and every value either represents a distance or is a reserved value that indicates that no distance data could be captured.

*Of course this is only true in the ideal case where the capturing method is noiseless, so a scene model that also includes a measure of probabilistic accuracy would be required to be truly independent of the capturing method.



(a) range image **(b)** ground truth segmentation



(c) metric point cloud

Figure 1.1: (a) A range image, visualized as a grey-scale image with darker values representing pixels that have larger depth and are therefore further away, along with (b) its ground truth planar surface segmentation and (c) a point cloud using metric distances that was created with a transformation function.

- There is some known transformation function that uses the depth value of a pixel along with its location in the raster image to determine the actual 3-dimensional position in space (relative to the vantage point).

By limiting this study to range images that satisfy these conditions we make explicit the assumptions about these images and clarify the consequences for how these images can be used. For instance, these assumptions imply that, while a range image can be converted to a point cloud with the appropriate scaling algorithm, the reverse is not generally true. An arbitrary point cloud can have points at any location in space, but a range image is taken from a single viewpoint and points cannot occlude one another (from this view). Also, since a transformation function is provided for scaling the range image to a point cloud, no assumptions need to be made about how the points are sampled, which allows methods that do not require the points in the point cloud to be sampled on a regular grid and that possibly use different sampling methods for each of the axes. It should be noted that in this definition the transformation function need not be invertible which could be problematic when a transformation from the point cloud back to the range image is required.

1.2.3 Obtaining range images

There are a variety of methods that can be used to obtain range images, and some are listed below.

- **Time-of-flight:** Similar to radar or LIDAR, a light pulse can be used to measure the distance to a point by measuring the time it takes for the light pulse to travel towards the point and reflect back to the sensor.
- **Structured light:** A light pattern can be used to illuminate a scene and the depth of the scene can be inferred from the way the known pattern is distorted from the viewpoint of a camera.
- **Stereo triangulation:** Two normal cameras can be used to capture a scene from two viewpoints and, if corresponding points can be found in both images, triangulation can be used to determine the distance to those points.
- **Interferometry:** Coherent light can determine the distance to a point by measuring the phase shift of the reflected light relative to the initial light source.
- **Synthetic:** If range images are required for testing purposes, they can also be synthetically generated using computer simulations or even the advanced graphic engines that are currently used for games and films.

Each of these methods has different advantages in terms of speed, complexity, robustness, physical limitations and cost. While stereo triangulation uses cheap hardware and provides passive sensing (i.e. no signal is transmitted), it requires very complex computations. Time-of-flight cameras or methods using interferometry can be very accurate but require costly hardware. Recent advances with structured light cameras mean that they are available at low cost but, depending on the light frequency used, they can suffer from environmental interference and limited range. The evaluation framework used for evaluation in this study has datasets available that use a laser range finder (using interferometry) and a structured light scanner.

1.2.4 Range image segmentation

Segmentation is the process of dividing an image into multiple regions according to some criteria. One of the most significant problems in designing segmentation algorithms in the general field of computer vision is that the concept of segmentation is not always well defined, which leads to many different segmentations being equally viable. This is clearly seen in the variation found when humans are tasked to manually segment colour images [1]. In the case of range image segmentation the problem becomes one of segmenting geometry instead of intensities and, as such, becomes less subjective.

Range image segmentation focuses on the separation of surfaces that describe the structure of the scene. Hoover et al. [2] compare a variety of sources that define segmentation and find no consensus on a formal definition. While the various sources differ only in small ways, they remain insufficient for creating a practical method to quantitatively evaluate range segmentations. A formal definition of segmentation fitting these requirements can be summarized as follows [3].

1. Every pixel in the range image belongs to a region.
2. Every region's pixels are 4-connected.
3. All regions are disjoint (no overlapping regions).
4. All pixels in a region belong to the same surface.

5. If two separate regions are adjacent then they represent different surfaces.
6. Noise regions are allowed where no valid measurement was possible, which all have the same label (violating rule 2) and for which rules 4 and 5 do not apply.

As can be seen, this definition makes special provisions for areas that contain no measurements as can often happen with some range imaging techniques.

1.2.5 Planar segmentation

The definition for a valid segmentation still leaves open the question of how each surface is defined. It is possible to use any number of different surface types, with each region being some type of surface with a set of parameters. A great advantage of using multiple surface types is that surfaces can be identified more specifically, which introduces more contextual information, and that large areas can be represented much more compactly using a single surface with a small number of parameters. However, while segmentations with multiple surface types simplify the contextual modelling of the scene, algorithms that produce these segmentations are often high in complexity.

The alternative is to use a single surface type, usually a plane (although sometimes a quadratic surface is used to represent planes and curved surfaces with a single model). While it might seem that real-world scenes are ill-suited for segmentation into planar regions, there is actually no significant loss in generality [4] as any surface can be linearly approximated to arbitrary accuracy, although at the cost of using many smaller surfaces. This is in fact already the established practice in computer graphics where polygonal mesh models are used. While planar segmentation approaches allow simpler processing, they might not always segment non-planar surfaces into planes in an appropriate manner. Even if they do, the numerous small surfaces that are created would likely be more susceptible to noise and distortion. In the end, from the findings made in evaluating planar segmentation approaches [2], and from the results of subsequent work, it can be concluded that planar segmentation is an unsolved problem that still requires improvement and is worthy of further investigation.

The objective of this study is to consider the current approaches used to solve the planar segmentation problem and to develop and test a new approach which will then be compared in a quantitative manner to other methods. In this study the problem is considered from a theoretical point of view with a focus on achieving good accuracy and robustness. The scope of the study does not include a strong focus on improving other practical considerations such as execution speed or the results for any specific application (although these considerations are not fully ignored). Now that the problem has been stated and motivations have been given for researching planar segmentation of range images, we can consider in more detail the related work that has been done in this field.

1.3 Related work

Within the field of range image segmentation, some methods developed to evaluate segmentation results will now be considered, and then a brief overview will be given of existing methods used to achieve these results.

1.3.1 Evaluating range image segmentations

A significant milestone in the field of range image segmentation was the creation of a test methodology for quantitatively evaluating results from different segmentation algorithms. This was done by Hoover et al. in their landmark paper: "*An experimental comparison of range image segmentation algorithms*" [2] which was published in 1996. In this work they strive to push the field of range image segmentation onto a sound experimental basis by proposing formal definitions of performance metrics that can be used to compare segmentation results against known ground truths so that algorithms can be properly compared to establish a state of the art. They also captured two range image datasets (called Perceptron and ABW) with commonly used hardware and range imaging techniques, and painstakingly defined ground truths for all images in these datasets. They also created a framework to automate the evaluation of any segmentation algorithm on their datasets and they then invited research groups to evaluate their algorithms with this framework. Their work includes the comparison results on both datasets using algorithms from research groups at four different universities. A continuation of their work was done by Jiang et al. and published in 2000 with the title of "*Some further results of experimental comparison of range image segmentation algorithms*" [5]. In this work three more methods were evaluated, although only one of the datasets was used (ABW). This entire framework and its performance metrics and datasets are discussed in more detail in chapter 6.

While the framework was created to be general enough to handle almost any type of segmentation, the datasets were created to assess only planar segmentation. This was done by choosing objects and scenes that contain only polyhedra and flat surfaces so that the segmentation algorithm need only consider planar surfaces. An extension to this work that includes datasets and ground truths for curved surfaces was presented in 1997 by Powell et al. in a thesis [6] and conference paper [7] both entitled "*Comparing curved-surface range image segmenters*". Some follow-up work was also done that allows approaches with many different parameters to be trained in an automated manner [3, 8]. The evaluation framework created by Hoover et al. was also used by various others to evaluate their own work, and will be used in this study to compare the results of these approaches against one another and our own in chapter 6.

1.3.2 Range image segmentation techniques

Most methods of range image segmentation can be categorized as either region-based segmentation or edge-based segmentation methods [9]. Region-based methods attempt to achieve a segmentation by assigning the same region label to clusters of pixels that have identical or similar properties*. Edge-based methods look for the differences between adjacent pixels that identify edges between regions and try to find a closed set of edges for each region. Although this classification is not absolute, methods that defy it are often either a hybrid of the two or not *surface* segmenters, but rather *object* segmenters that directly attempt a full scene reconstruction instead of a segmentation (but whose results can then be used to determine a segmentation). More detail and specific examples of methods falling into each of these categories are discussed throughout chapters 2 and 3.

1.4 Our contribution

The principal substance of the contribution of this study to the field of range image segmentation is the development and testing of a novel segmentation algorithm based on graph cut

*Although some sources use the term 'region-based methods' to refer only to methods using iterative region growing, our use of the term includes any type of clustering method.

optimization. While this study continually discusses many approaches and related work within the field, there is greater focus on the parts that are integrated into this new algorithm. An overview of our segmentation approach is illustrated in figure 1.2. Both region properties, in the form of surface normals, and edge properties are incorporated into this approach, and while this study's use of graph cut optimization delivers a type of normal reconstruction, some simple post-processing is then performed to transform those results into a segmentation.

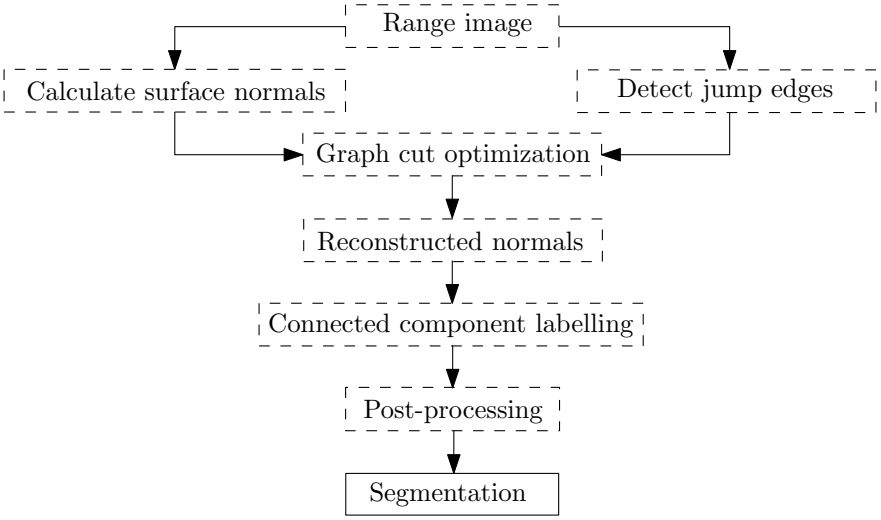


Figure 1.2: An overview of the approach followed by the range image segmentation algorithm proposed in this study. Solid lines indicate data results and dashed lines indicate processes that transform the data.

1.5 Thesis outline

Methods for detecting edges in range images are discussed in chapter 2 along with segmentation approaches that are strongly based on using the edge data (edge-based methods). The calculation of surface normals is discussed in chapter 3, along with a number of region-based approaches that segment range images by considering various regions properties. Other region transformations that are used, such as connected component labelling and morphological operations, are also covered therein. Along with the region-based approaches, global optimization approaches are also considered and used to introduce the concept of graph cut optimization. Following that, chapter 4 goes in depth into the theory of graph cut optimization, which is then useful for chapter 5 where we detail all the considerations of how graph cut optimization is used for the purpose of range image segmentation. In chapter 6 it is then considered how to analyse the segmentation results of the proposed approach by considering the evaluation framework of Hoover et al. [2], which is then used to evaluate the approach and compare it against other algorithms. This study finishes with chapter 7 where we impart our conclusions about the proposed approach and the current state of range image segmentation.

Chapter 2

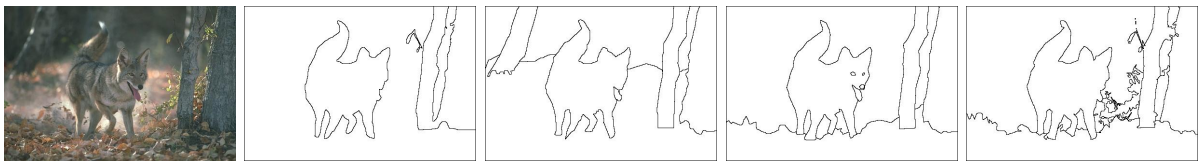
Edge-based segmentation

Edge-based segmentation methods focus on detecting the edges between surfaces and then completing the segmentation using these edges as a basis. This usually entails the use of operations that find edge candidates by looking at local regions around pixels. Due to noise these local operations rarely find a complete set of edges and some type of contour closure method is then often used. The local nature of the operations means that these approaches are generally much faster than the region-based alternatives, but often also less robust to noise.

In general, edge-based methods have been less prevalent in the literature than region-based approaches for range image segmentation [10]. This is likely due to the success of region-based methods in the segmentation of colour images when compared to edge detection approaches. Edge detection can be challenging in colour images as disparities in pixel values can be caused by numerous different physical processes such as occlusions, shadows, reflections and textures properties. Finding which disparities are caused by the actual edges of objects can therefore not be done by looking only at the local properties of the image. This is even more of a problem since the segmentation of colour images is not a well-defined concept and may have more than one "correct" answer for a given image. This is substantiated by the large variations found when humans are tasked to segment images manually [1], an example of which can be seen in figure 2.1.

Fortunately, these problems do not all translate to range images where segmentation regions are more strictly defined as specific physical and geometric surface types. Recent techniques that were developed specifically for range images have the potential to outperform many region-based approaches in both efficiency and accuracy.

The rest of this chapter goes into more detail of the types of edges found in range images and the edge detection theory and techniques used to find them.



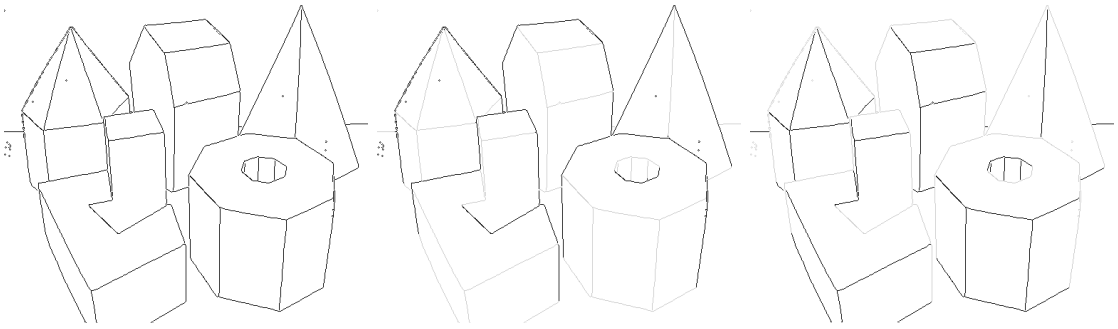
(a) **(b)** **(c)** **(d)** **(e)**

Figure 2.1: A colour image and some manually segmented versions. This example illustrates the fact that, for natural images, a "correct" segmentation is an ill-defined notion. Image source: [1].

2.1 Edge types

Edges are caused either by surfaces obstructing one another or when surfaces intersect. This can be seen in figure 2.2 where the edges of a range image have been separated by these causes. For edge detection, edges are often considered at a lower level by looking at a cross-section of a range image, such as in figure 2.3. Edges are then be divided into three types, as illustrated in figure 2.4. Jump (or step) edges are formed by discontinuities in the depth data due to occluding surfaces. Crease edges are discontinuities in the surface normals (discussed in depth in section 3.1.2) that occur where two surfaces intersect with each other. Smooth edges are specific to non-planar surfaces as they are discontinuities in curvature. Since this study considers only planar surfaces, it will not be considering approaches for finding smooth edges.

Crease edges are further subdivided as being either roof or ramp (also called non-roof) edges, each being either concave or convex [11, 12]. Figure 2.5 gives examples. Roof edges are local extrema that have either higher or lower depth values on both sides, while ramp edges have lower values on one side and higher values on the other. Unlike the properties of concavity or



(a) all edges **(b)** edges caused by surfaces occluding other surfaces **(c)** edges caused by surfaces intersecting with each other

Figure 2.2: The edges from a range image separated by physical cause.

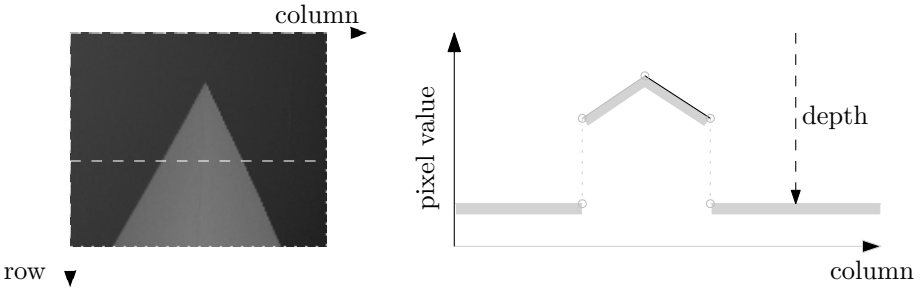
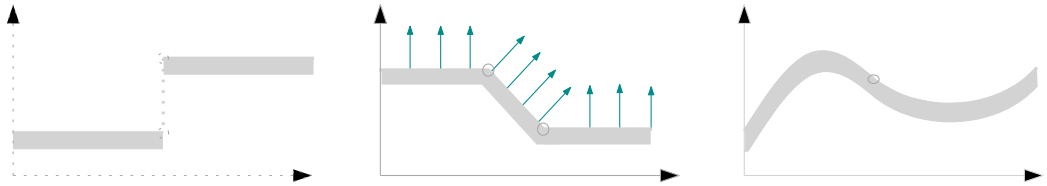


Figure 2.3: An idealized cross section of a range image allows us to consider the different types of edges.



(a) jump edge **(b)** crease edges **(c)** smooth edge

Figure 2.4: The different types of edges in range images correspond to discontinuities in depth, discontinuities in normals, and discontinuities in curvature.

convexity, the concepts of roof and ramp edges are not viewpoint-invariant as a rotation can alter an edge between the two, making the classification less useful.

Figure 2.5: The subtypes of crease edges are: (a) convex roof edge, (b) concave roof edge, (c) convex ramp edge, (d) concave ramp edge.

When detecting edges in range images, there are two dominant strategies. The first is image gradient detection which is a data-based approach that looks for changes in the values of the image data. The second alternative is an approach using scan-line approximation that is a top-down model-fitting approach where an image is cut into scan-lines that are then divided into segments whose separation points identify possible edges. These techniques are discussed in the following sections.

2.2 Image gradient edge detection

Image gradient edge detection was developed for processing colour images and when applied in the context of range image segmentation is still of use for detecting jump edges, but is not well suited to detecting crease and smooth edges. This is because finding crease and smooth edges in this manner would require looking for changes in the surface normals and surface curvatures, which are not directly available and need to be inferred from the depth data. This is complicated as any noise in the depth data not only propagates to these properties but is also highly amplified. Techniques following this approach for crease and smooth edges thus focus on robustly estimating these properties in the presence of noise, which will be discussed in more detail in the next chapter.

Jump edges can be found as significant local changes in the depth data, which correspond to large derivatives of the depth values. If the depth values d are represented by a function f in two variables (the axes of an image), as

$$d = f(x, y), \tag{2.1}$$

then the gradient at a particular point (x, y) is a vector containing the partial derivatives,

$$\vec{\nabla} f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} g_x \\ g_y \end{bmatrix}. \tag{2.2}$$

This vector has a magnitude of

$$M(x, y) = \|\vec{\nabla} f\| = \sqrt{g_x^2 + g_y^2}, \tag{2.3}$$

and orientation of

$$\alpha(x, y) = \text{atan2}(g_y, g_x). \tag{2.4}$$

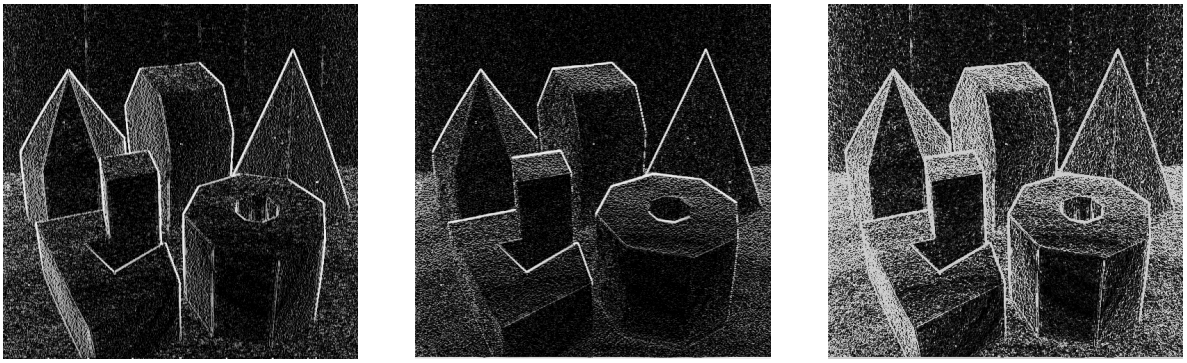
A derivative, which requires a continuous function, cannot be calculated when working with discrete values and it is rather estimated from the change in values between adjacent pixels. There are a variety of ways to estimate the derivative for a discrete image, a common one being the use of Sobel operators.

2.2.1 The Sobel operators

The Sobel operators [13] are linear spatial filters that estimate the image gradient at a point by considering the 3×3 area surrounding the point. The image is convolved with the kernels

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{2.5}$$

separately to obtain two images G_x and G_y whose values give the gradient at every pixel. The theoretical basis for these kernels is actually quite simple. It combines the simplest symmetric kernel that can estimate an image gradient, $[-1, 0, 1]$, with the $[1, 2, 1]$ kernel which acts as a weighted averaging filter. This averaging kernel allows the operator to be slightly more robust to noise. The results of the kernel operations are then combined according to equations (2.3) and (2.4) to produce the edge magnitude and orientation at every discrete location in the image. An example is shown in figure 2.6.



(a) G_x **(b)** G_y **(c)** $M(x, y)$

Figure 2.6: Grey-scale images that correspond to the magnitude of the (a) horizontal and (b) vertical components of the gradient, as well as (c) their combined magnitude. The input range image is the one shown in figure 1.1(a).

2.2.2 Thresholding edge strengths

Once the image gradient magnitude has been determined for each pixel in the image, it is then necessary to determine which pixels will be classified as edges. This is often done by simply choosing a threshold value and pixels that have magnitudes which are larger than the threshold are chosen as edges. While it is clear that pixels with the largest magnitude are more likely to be edges, it is often impossible to select a single threshold that will identify all edges while avoiding false edge responses. The value of a gradient magnitude is quite abstract and can vary significantly according to the input depth data, unless it has been normalized in some manner. It is therefore usually necessary to change the threshold for different images or even within the same image.

One thresholding technique, that is commonly used to choose a threshold on a per-image basis, is to use the average of gradient strengths across the entire image to determine a threshold value for that image. This is based on the idea that the mean of the gradient magnitude squared is roughly proportional to the signal-to-noise ratio (SNR) and that the cut-off threshold should be the root-mean-square (RMS) of this estimate [14], as

$$T = \sqrt{4 \cdot \text{mean}(g_x^2 + g_y^2)}. \tag{2.6}$$

The problems with using a single threshold for an entire image become especially clear in the common case where edge detection methods can generate strong candidate edges that are parallel and adjacent to actual edges. This type of multiple response is quite typical and usually the entire group of adjacent pixels are classified as edges, which leads to 'thick' edges with localization inaccuracies. Sometimes these thick edges are handled in a post-processing operation such as morphological skeletonization, although a more accurate option might be to take only the 'ridge peaks' of the magnitude response. Other possible problems such as impulse noise in the depth data can also lead to edges surrounding these noisy pixels. Solutions to these problems depend on the application and on which type of errors are observed. A solution that uses a different threshold value for different areas of the image is called an adaptive thresholding technique (or sometimes dynamic, variable or local thresholding [13]). The results of thresholding and ridge thinning are compared in figure 2.11 towards the end of this chapter.

2.2.3 Smoothing

Other options for removing some of the problems faced with thresholding are to avoid errors in the gradient magnitudes by altering the raw depth data and reducing the effects of noise. Handling noise in a general sense is a complex problem because noise can be unpredictable, can be caused by numerous sources that introduce different types of noise and, in cases where noise overwhelms the fine structure of an image, can cause information to be irretrievably lost.

Noise can often be described adequately by a Gaussian distribution, and the randomness becomes easier to handle once some assumptions are made such as that the noise is statistically independent at each pixel, that it has a zero mean and that it has a higher frequency and smaller amplitude than the overall signal. The noise can then be handled to some degree by smoothing out the data. While linear spatial filters can be used for averaging and edge detection, these methods are not usually as effective as some nonlinear filters for handling noise in an image. The problem is that averaging filters may over-smooth edges in the data. Comparisons have shown [15] that nonlinear filters such as the median filter can provide good edge-preserving smoothing, as can be seen in figure 2.7.

2.3 Optimal edge detection

Up to now this chapter has considered edge detection methods that were originally created with intensity (or colour) values in mind. While these methods do provide useful approaches for detecting jump edges, they do not provide any way to detect other edge types in a direct manner. Also, the physical properties that are represented by the values in a range image are affected by noise and quantization processes in slightly different manners that are not always handled well using these approaches. The work of Jiang and Bunke [11] provide a theoretical definition of an edge model that is tailored specifically to range images, and which is further discussed here.

In the 1-D version of this edge model (depicted in figure 2.8), the local area on each side of a pixel is modelled by a straight line function, and an edge occurs whenever the functions $f_1(x)$ and $f_2(x)$ are not the same. The functions are then evaluated at x_0 and the distance h between them is found with $h = |f_1(x_0) - f_2(x_0)|$. An edge is then classified as a jump edge when $h > 0$ and a crease edge otherwise. This model allows edge strengths to be assigned for each type of edge according to the physical properties of the edge, h for a jump edge and the difference between the normals of f_1 and f_2 for a crease edge.

When extending this model to 2-D the edge point now becomes (x_0, y_0) and its local environment is modelled by two planar patches, $f_1(x, y) = a_1x + b_1y + c_1$ and $f_2(x, y) = a_2x + b_2y + c_2$. The

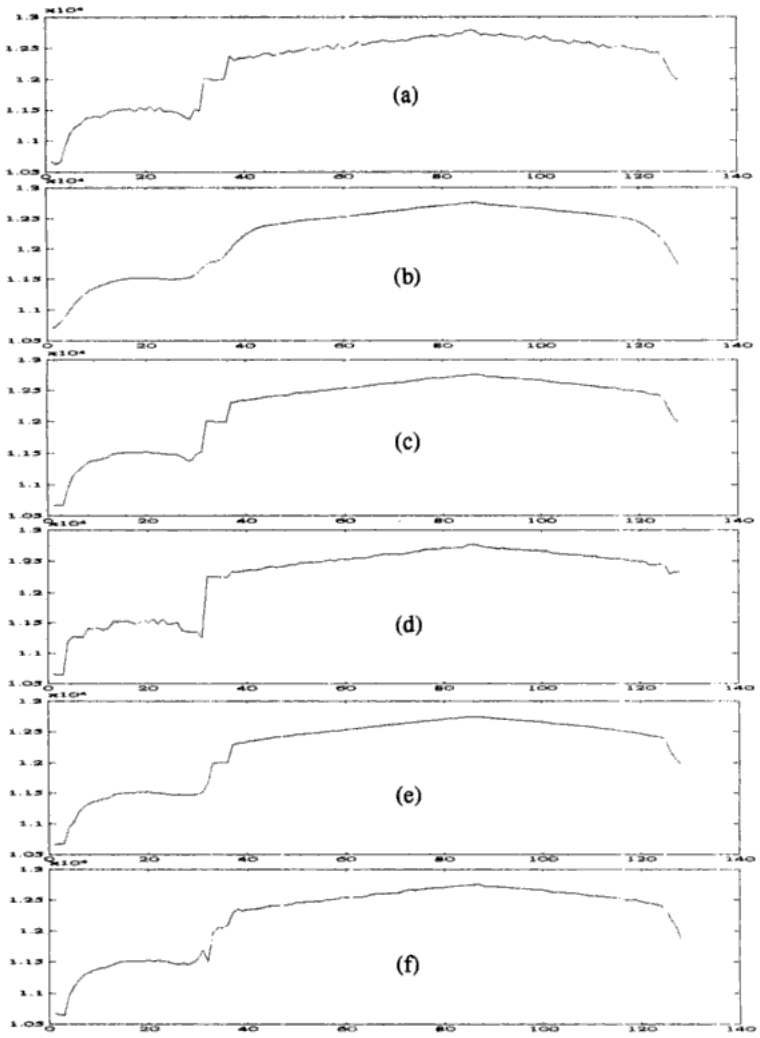


Figure 2.7: A comparison of various smoothing filters. The original noisy signal (a) is shown along with the results of (b) a Gaussian smoothing filter, (c) a median filter, (d) a k-NN filter, (e) the IRLS variable order procedure and (f) LTS with a second order model. Image source: [15].

edge strength for a jump edge is then defined as

$$|f_1(x_0, y_0) - f_2(x_0, y_0)|, \tag{2.7}$$

and for a crease edge as

$$\arccos \frac{(-a_1, b_1, 1) \cdot (-a_2, -b_2, 1)}{\|(-a_1, b_1, 1)\| \|(-a_2, -b_2, 1)\|}. \tag{2.8}$$

The work of [11] then also defines an optimal edge detector as one that supplies this crease edge strength correctly. Crease edges are generally much harder to find than jump edges and this definition for a crease edge strength is independent of the position and orientation of the scene relative to the range scanner, as well as invariant to changes in the coordinate system.

Modelling with planar patches does not exclude curved surfaces as small local regions can be approximated reasonably well with linear models. Other advantages of this edge model are that:

- edge strengths now have a direct geometric interpretation that somewhat simplifies thresholding choices;
- edge points can be classified into edge types;

- edge detection techniques can be evaluated according to deviation from the ideal edge strength;
- it can also handle range images that are not sampled on a regular grid;
- it excludes pixels on a highly sloped surface from being identified as edges.

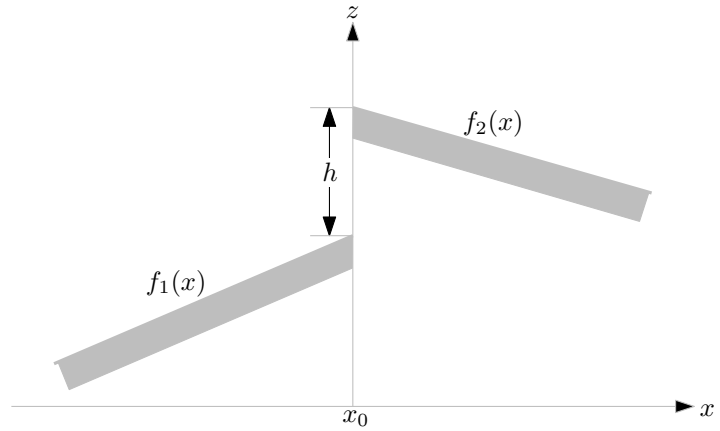


Figure 2.8: 1-D edge model.

2.3.1 Edge detection by scan-line approximation

The definition of optimal edge detection does not specify how these edges should be found or determined, and Jiang and Bunke then go further and develop a method using scan-lines to detect edges based on this edge model [11]. Their work is also used in [16] and slightly adapted in [17]. The method proceeds in two steps. First they apply an approximation method to scan-lines in the horizontal, vertical and two diagonal directions of the image. Each scan-line is approximated with a set of polynomial functions. Edge points are identified as the partitioning points between these functions, and the second step of their method is to combine the results of these four processes.

Scan-line approximation is performed by a split-and-merge algorithm (also known as the Ramer-Douglas-Peucker algorithm [18, 19]) that starts by performing a quadratic fit to the midpoint and two endpoints of the scan-line. The largest error $e_{\max}$ between the approximation function and the actual values is then found, and if the error is greater than some threshold value ϵ the scan-line is split at that point and the process is repeated recursively for each segment. This continues for all segments until no segment contains an error $e_{\max}$ that is larger than ϵ. An example is provided in figure 2.9.

This splitting method tends to produce spurious segments and is therefore sometimes followed by a merging step, although that is generally unnecessary as the edges created by the spurious segments will still have small edge strength values, and they should be filtered out during the thresholding stage. Merging can also be added to handle impulse noise that would cause tiny segments to be created. While this method is simple, Jiang and Bunke show that it is very robust to noise (unless of course the threshold value ϵ is too small in comparison to the amount of noise that is present).

Edge strengths are then calculated in a similar manner as in the definition of the 1-D optimal edge detector. In comparing this approach to the optimal edge detector's crease edge strengths, it can be seen that the accuracy is highly dependent on the specific crease edge and the approach angle that is necessary to find its maximum angle when traversing along a 1-D scan-line. This is illustrated by figure 2.10 where the same crease edge results in different angles depending on

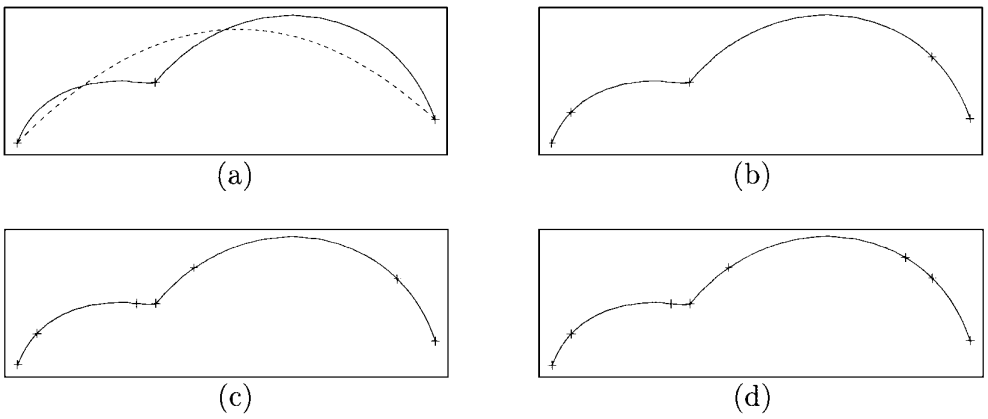


Figure 2.9: The various steps in the scan-line approximation of a curve. Image source: [11].

the angle of approach. For this reason scan-line approximations are performed along horizontal, vertical and both diagonal angles, as more angles of approach should improve the accuracy of the crease edge strengths.

The second phase of the approach is to use the edge strengths from the four approximation processes to determine which pixels are valid edges. The crease edge strengths from the four different angles are combined by choosing the largest value found at each point. An additional step is then also added to improve the localization of the edge points (which can often be off by a pixel or so). This step reconsiders the parameters of adjacent function segments, performs a least squares approximation along all the points of a segment (as opposed to only the mid- and endpoints as was done previously), and then back-projects the intersection point for these new functions to better determine the location of the edge point.

Up to now, the method used the strengths of a model-based approach to handle the noise on a low level. These results are then used to achieve a segmentation by also adding a final thresholding step to determine edge points, and then performing contour closure to achieve a viable segmentation*.

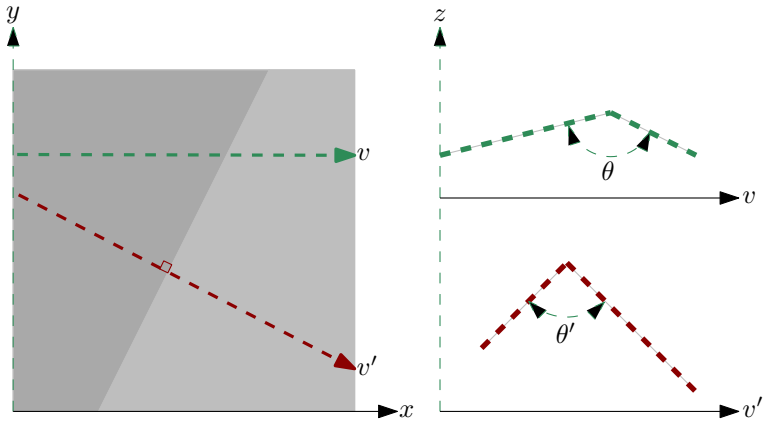
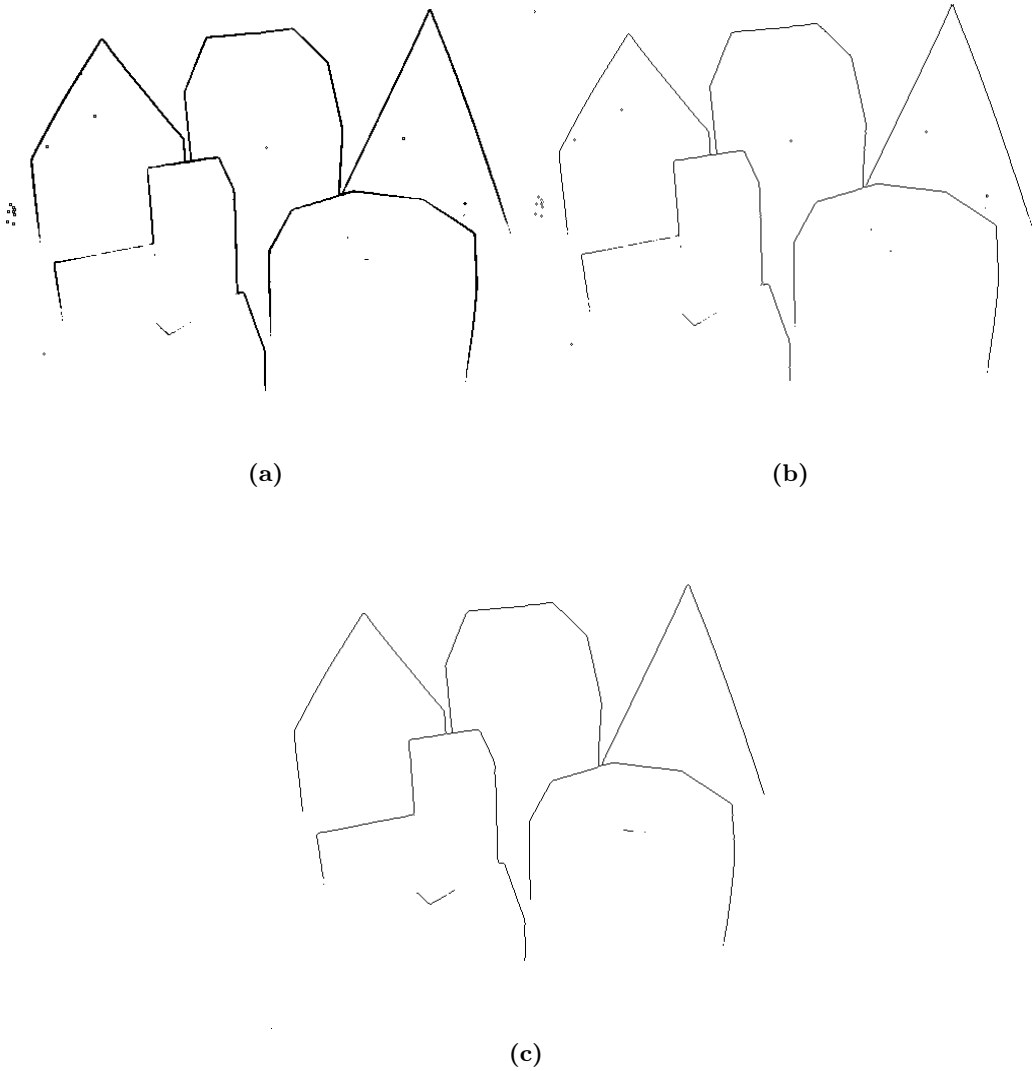


Figure 2.10: A comparison of how the choice of scan-line affects the crease edge strength obtained, and why the optimal edge strength is hard to obtain.

*It should be noted that an earlier version of this approach [20, 21, 22] finds the line segments and then uses region growing with these line segments to achieve a planar segmentation, instead of finding edges and using contour closure. That approach is also evaluated in [2] where it is called the UB algorithm (in reference to the University of Bern in Switzerland where it was developed).

2.4 Contour closure

Both the techniques of image gradient detection and scan-line approximation either provide many possible edges or are thresholded to provide only some definite edges. Unfortunately they both have no guarantee of providing a closed set of edges which is required for a segmentation and most likely have gaps in the edge data. Contour closure refers to any of the techniques that are used for generating a closed set of edges from a sparse set of edge points. The contour closure technique required depends on how reliable the edge detection approach is. In the case where edges are found reliably and with only a few pixels disconnecting the edges, a simple morphological operation like bridging can be used to complete the contours. An example is shown in figure 2.11(c).



(a) **(b)**

(c)

Figure 2.11: The Sobel operator, combined with the threshold described in equation (2.6), produced the result in (a). In (b), ridge thinning was used to improve the results by better localizing the edges. For (c), the same technique was preceded by a median edge filter and a simple morphological bridge was used for contour completion.

In other cases, more advanced techniques are required such as fitting lines or curves and using a hypothesis generation and verification approach to guess where edge pixels might be missing [10, 11]. In [23] a contour closure technique is presented which is based on the Delaunay triangulation

of the edge points. The triangulation is then used as a graph from which the minimum spanning tree is calculated and then combined with some post-processing to present a closed set of contours. The problem of contour closure is approached in a different manner in [24] where the image is first broken up into superpixels and closed contours are then found by grouping subsets of superpixels whose collective boundaries indicate strong edges. Other notable approaches to contour closure include ratio contours [25, 26] and other edge grouping approaches [27, 28].

In cases where the edge data is too sparse though, it can be impossible to make any further conclusions. For contour closure our segmentation technique (discussed in chapter 5) requires only morphological operations which are discussed in the following chapter.

Chapter 3

Region-based segmentation

Region-based segmentation is more commonly associated with region growing methods, although with range image segmentation the term can also be used to include parametric model-based segmentation [29]. A disadvantage that is commonly shared by region-based segmentation techniques is that their results are often noisy along edges between regions, although these problems can be alleviated by using hybrid systems that also incorporate edge information.

This chapter covers some background and then discusses commonly used region-based segmentation techniques. It then gives special focus to the concept of global optimization and how it can be, and has been, applied to help solve the range image segmentation problem.

3.1 Background

A variety of related concepts and methods that are useful tools for these segmentation approaches are first discussed before covering the various region-based segmentation methods. These include some binary image operations that are useful for manipulating region shapes and labelling regions, and also the definitions of surface normals and curvature which are locally determined properties that can be used to characterize regions in range images.

3.1.1 Binary image operations

A binary image is an image where each pixel can have only one of two values, and is commonly visualized as a black and white image. Binary images are considered because any single region in an image can be represented with a binary image, with separate values for the region and the rest of the image which will respectively be referred to as the foreground and background. Many operations performed on individual regions are therefore based on approaches developed for binary images. It is for these reasons that the operations of mathematical morphology and connected component labelling are briefly discussed.

Mathematical morphology

Mathematical morphology includes a number of set operations that manipulate an image by probing it with some structuring element [13]. The structuring element is some small binary image (usually 2×2 or 3×3) that is moved across an image and used to perform a calculation to select a new value for each pixel in the image. The calculations can be bitwise operations between the image and structuring element, or in fact any linear or nonlinear calculation. Because

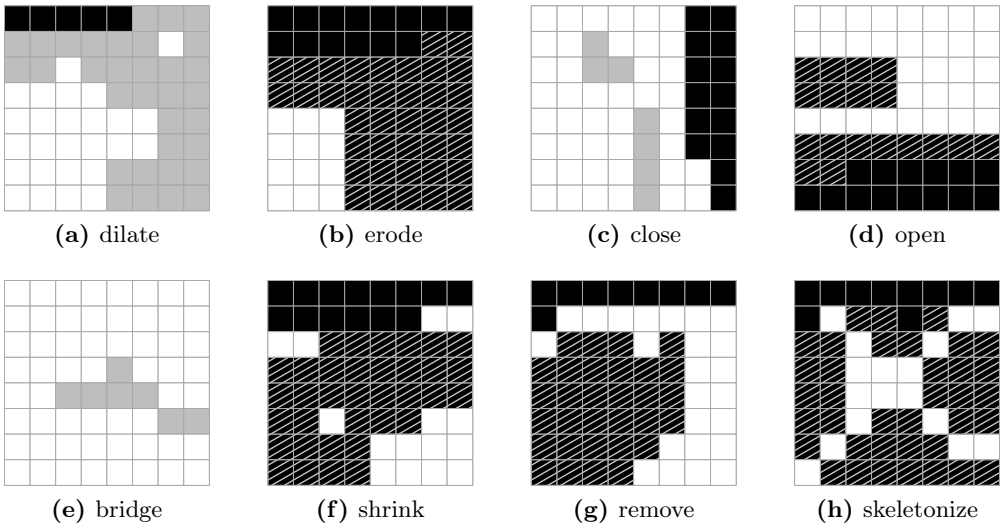
of this, a general framework for implementing morphological operations is often implemented by using a look-up table that defines the outcome value of a pixel for each specific possibility of configurations in its neighbourhood (whose size and shape are that of the structuring element). Morphological operations are often applied multiple times until the image stabilizes (i.e. no more changes occur in the image). There are also a variety of ways for how these operations handle the borders of images, such as assuming that everything outside the image is only background or that it mirrors the image content. For further discussion it will be assumed that an operation is performed only on the foreground of the image.

The basic binary morphological operations are dilation and erosion. These operations are duals of each other, as an erosion on the foreground is equivalent to a dilation on the background (assuming identical structuring elements). The dilation operation expands regions, as pixels close to region boundaries are changed to foreground pixels. Erosion contracts regions by removing foreground pixels that are on or near region boundaries.

Other common morphological operations are shown in figure 3.1, and these include:

- *opening*, which can be implemented as an erosion followed by a dilation, will remove small regions;
- *closing*, which can be implemented as a dilation followed by an erosion, will remove small holes;
- *bridging* connects regions separated by small gaps;
- *shrinking*, when performed until stabilization, turns regions without holes into points and regions with holes into connected rings around those holes;
- *skeletonization* is usually applied until stabilization and removes pixels on boundaries without breaking regions apart;
- *removing* removes all the interior pixels of a region.

The scale of application for each of these operations depends on the size of the structuring



(a) dilate **(b)** erode **(c)** close **(d)** open

(e) bridge **(f)** shrink **(g)** remove **(h)** skeletonize

Figure 3.1: Morphological operations performed on various binary images. White pixels are foreground, black pixels are background, grey pixels are foreground pixels that have been added by the morphological operation and shaded pixels have been removed by the operation. The results around the borders of the image can be handled in multiple ways, although it does not apply to these examples as they are subregions of a larger image.

element used. As can be seen, these and other morphological operations provide a large variety of methods for manipulating various region properties.

Connected component labelling

Connected component labelling is a method used to label regions in a binary image. It is useful because the same method is easily extended to relabel a segmentation result that contains disconnected regions that should be separately labelled. There are two variants of the algorithm: recursive and sequential. The recursive approach is parallelizable and proceeds by a process similar to a simplified version of region growing which is discussed later. The sequential approach is useful because of its efficiency as it requires only two passes over the whole image to label all regions. The first pass assigns temporary labels and keeps track of possible errors, while the second pass corrects any errors. It proceeds as follows.

1. Scan through image pixels from top-left to bottom-right.
   (a) If top and left neighbours are in the same region as this one, but were assigned to different labels, then assign the top label to this pixel and enter the labels into an equivalence table as equivalent labels.
   (b) Otherwise, if both neighbours are in the same region with the same label, assign their label to this pixel.
   (c) Otherwise, if only one of the left or top neighbours is in the same region, then assign that label to this pixel.
   (d) Otherwise, assign a new label to this pixel (and add it to equivalence table).
2. Find the lowest label for each equivalent set in the equivalence table.
3. Rescan the image and replace each label by the lowest label in the equivalence table.

3.1.2 Surface normals

Surface normals represent much of the orientation and curvature information of a surface and are thus one of the most useful region properties that can be calculated for a range image. They also provide the most direct way of identifying crease edges and for grouping points belonging to the same planar surfaces.

Normal definition

A line or a vector is called a normal of some object when it is perpendicular to that object. In the case of plane curves the normal at any point on the curve is a line perpendicular to the tangential line at that point. Note that because of this, the normal of a curve is directly related to its first derivative, which can be used to determine the tangential line. This relationship is useful in the case of a discrete function where determining a normal is not as obvious, but finite difference methods can be used to estimate the derivative. Figure 3.2 shows an illustration of how normals are calculated for a continuous curve and some sampled versions of it.

A 3-D normal vector consists of an $[x, y, z]$ Cartesian coordinate triplet that indicates where the end of a normal would be if it were placed at the origin of the axis system, $[0, 0, 0]$. Normals can also be represented in a spherical coordinate system as $[r, \theta, \phi]$, where r, θ and ϕ respectively

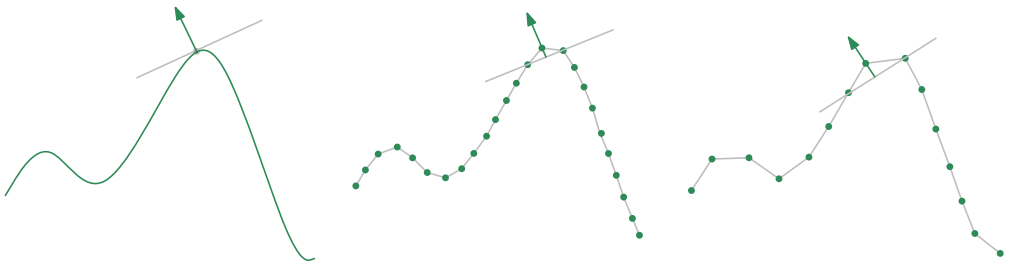


Figure 3.2: The normal for a continuous curve, and for the same curve discretized. Note that the curve is sampled at regular intervals along the curve instead of regularly along an axis as happens in range images, in which case the degradation can be much worse.

refer to the normal's magnitude, azimuth and elevation. Although most common conventions* use a polar angle (measured from the z-axis), this study follows the convention of using an elevation angle (measured from the x-y plane). We also choose to define surface normals to point towards (and not away from) the image sensor. In accordance with the coordinate system used with our range images, this means that the normals are oriented towards the negative z-axis. The convention that will consistently be used is illustrated in figure 3.3. When working with range images where the surface normals represent the orientation of a surface, the magnitude of the normal vector can also be ignored for most purposes and the vector can be normalized by dividing the vector by its magnitude:

$$\frac{\vec{v}}{\|\vec{v}\|} = \begin{bmatrix} \frac{x}{\sqrt{x^2+y^2+z^2}} \\ \frac{y}{\sqrt{x^2+y^2+z^2}} \\ \frac{z}{\sqrt{x^2+y^2+z^2}} \end{bmatrix}. \tag{3.1}$$

In this case the spherical coordinate will always have a magnitude of $r = 1$ and when using this system with normalized vectors the magnitude can be dropped and normals represented as an azimuth-elevation vector,

$$\vec{v} = \begin{bmatrix} \theta \\ \phi \end{bmatrix}, \tag{3.2}$$

where $0 \leq \theta < 2\pi$ and $0 \leq \phi \leq \frac{\pi}{2}$. Note that orienting all the normals towards the image sensor means the space of possible normal vectors is equivalent to points on the surface of a unit hemisphere.

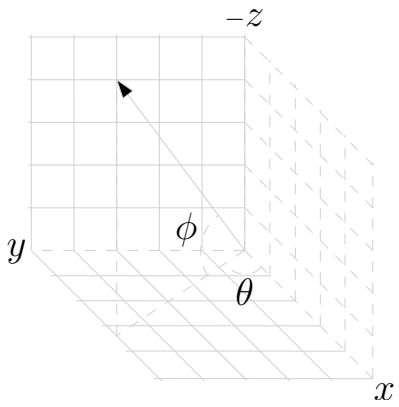


Figure 3.3: The spherical coordinate system used for surface normals in this study. Note the negative z-axis as surface normals are orientated to point towards the image sensor.

*Spherical coordinate systems follow a variety of coordinate conventions. Common conventions in mathematics use θ and ϕ as the azimuth and polar angles, while in physics this is generally swapped.

There are a variety of ways to determine the normal of a surface function. For a plane given by the equation

$$ax + by + cz + d = 0, \tag{3.3}$$

the (unnormalized) surface normal is the vector

$$\vec{v}_n = \begin{bmatrix} a \\ b \\ c \end{bmatrix}. \tag{3.4}$$

If the equation of a planar surface is unknown, the normal can be calculated from three points on the surface,

$$\vec{p}_1 = [\ x_1 \quad y_1 \quad z_1\ ]^{\mathrm{T}}, \quad \vec{p}_2 = [\ x_2 \quad y_2 \quad z_2\ ]^{\mathrm{T}}, \quad \vec{p}_3 = [\ x_3 \quad y_3 \quad z_3\ ]^{\mathrm{T}}. \tag{3.5}$$

Two vectors are constructed according to

$$\begin{aligned} \vec{v_1} &= \vec{p_1} - \vec{p_2}, \\ \vec{v_2} &= \vec{p_2} - \vec{p_3}, \end{aligned} \tag{3.6}$$

and their cross-product will then give a vector perpendicular to that plane (i.e. a surface normal):

$$\vec{v}_n = \vec{v}_1 \times \vec{v}_2. \tag{3.7}$$

This works for any three unique points as long as $\vec{v}_1$ and $\vec{v}_2$ are not parallel to each other. This method is also used to calculate the normal for any triangle, which is useful when the points in a range image have been triangulated. Furthermore, it remains useful for non-planar surfaces since the local area around a point can be approximated with a plane. The accuracy of the calculated normal then depends on how close the chosen points are to one another (which is limited by the sampling resolution) and how extreme the curvature of the surface is at that point.

Unfortunately, since range images provide only a noisy sampling of a surface, these calculations are not always applicable and other methods are required. These methods need to be quite robust as small amounts of noise in a range image can disrupt normals to a much larger extent, as illustrated in figure 3.4. The sampling process also destroys the orientation and curvature information and it only resides implicitly in the relationships between sampled points and their neighbours. There are a variety of methods that have been developed to estimate the surface normals, all of which consider a point along with its neighbours.

Neighbourhood structure

The neighbourhood of any point can be defined in a few different ways that are useful for the estimation of the normal. This neighbourhood structure is often formalized as an undirected graph where vertices correspond to points, and neighbouring points are connected by edges. There are primarily two types of graphs that take advantage of the point cloud nature of range data.

- A k nearest neighbour graph connects each point to the k nearest points as measured by the Euclidean distance between points. The number of neighbours is thus fixed and a triangulation of this graph structure may have overlapping triangles.
- Delaunay tessellation connects points into a tetrahedral mesh such that no points lie within the circumsphere of any tetrahedral. The number of neighbours is thus variable, but a triangulation will not have overlapping triangles. To estimate normals of a surface the tetrahedral mesh still needs to be simplified to a triangular mesh, as mentioned in [30].

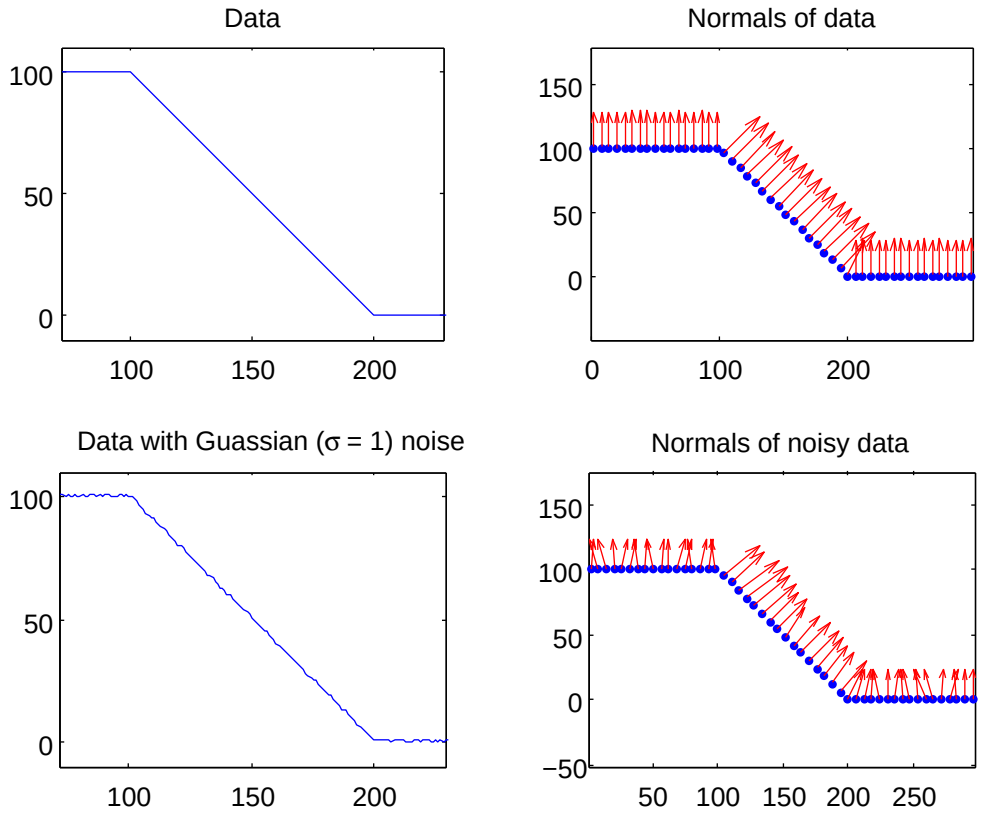


Figure 3.4: An example of how normals have a large reaction to small amounts of noise.

Three different neighbour graph structures that take advantage of the natural grid structure of range images are also commonly found.

- The pixel connectivity inherent in the grid structure can be used directly to determine neighbours. This usually refers to 4-connectivity where neighbouring points are those that lie above, below, to the left and right in the grid structure, although 8-connectivity, that also includes diagonal points as neighbours, is also an option.
- An $n \times n$ window centred on the point can also be used to create a larger neighbourhood size within the grid structure. The value of n is usually relatively small and is always odd, as even values would lead to an asymmetric neighbourhood. A 1×1 window would imply no neighbours and a 3×3 window is equivalent to 8-connectivity.
- Delaunay triangulation is the analogue of Delaunay tessellation when performed in 2-D. It connects points into a triangulation such that no points lie within the circumcircle of any triangle.

While most normal estimation methods are compatible with any of these methods, some require a minimum number of neighbours that might not be met with the variable number of neighbours found with the Delaunay methods. While having a large number of neighbours is beneficial to cancel out the effects of outliers and noise, it is often avoided as it leads to unintended averaging and smoothing effects. The choice of which method is best is otherwise not easily determined and is closely tied to which normal estimation method is used, while also being influenced by which options are available due to the specific data structures available in an implementation.

Normal estimation

The work in [30] compares methods for normal estimation and finds that these methods can be divided into averaging and optimization-based methods.

Averaging methods estimate the normal at a point by taking a weighted average of the normals for the triangles created by the point and neighbouring pairs of points. The differences in these methods lie in how the normal for each triangle is weighted and examples include area-weighted, angle-weighted, centroid-weighted and gravitational-weighted.

A number of seemingly different techniques can be classified as optimization-based methods that differ only in their cost function. These methods estimate a normal by optimizing a cost function which penalizes some criteria, e.g. minimizing the distance of the points to some local plane or maximizing the angle between the normal and tangential vectors (as illustrated in figure 3.5). Methods that also estimate the curvature, and not just the orientation, fit quadric surfaces instead of planes. Methods that minimize the fitting error often use least squares (LS), while methods that minimize the variance often use principal component analysis (PCA)*. Many variations are found that try to lessen the effects of outliers or the inaccuracies that are found near jump edges, usually by trying to explicitly detect and then exclude them. These problems can also be handled with robust estimators (discussed later), although their computational costs can be excessive for this purpose.

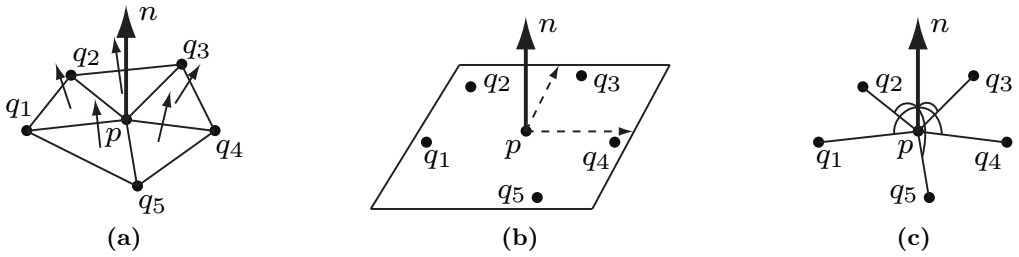


Figure 3.5: Different normal estimation approaches: (a) normals of neighbouring triangles are averaged, (b) a plane is fitted to the point and its neighbours and (c) the sum of angles between tangential vectors and the normal is maximized. Image source: [30].

3.2 Segmentation approaches

A variety of region-based segmentation approaches, that have been used for range image segmentation, will now be discussed.

3.2.1 Region splitting

Region splitting is a top-down segmentation approach that progressively subdivides an image into smaller regions until all regions are homogeneous as defined by some chosen metric. The general approach is as follows.

1. The entire image is considered the 'region of interest'.
2. If the current 'region of interest' satisfies some similarity constraint, then:

*Both methods, along with any linear minimization problem, can be solved by using the singular value decomposition (SVD).

(a) the region is completed and the 'region of interest' changes to a region that has not yet been considered;

(b) otherwise, the region is divided into subregions and (if the subregions are larger than a single pixel) a subregion is chosen as the new 'region of interest';

(c) if all regions have been completed proceed to step 3, otherwise proceed to step 2.

3. All regions should be homogeneous (with the worst case being regions the size of a single pixel).

This approach may introduce inaccuracy around the borders of regions, and many adjacent regions can have similar properties. This is because it is unlikely that the chosen method of region subdivision divides a region into subregions that precisely align and match the shape of correct regions. To handle this, a step is often introduced that then considers whether adjacent regions should be merged again. The method is then referred to as a split-and-merge algorithm. One approach of dividing regions is to use quadtree subdivision where regions are divided into equal quarters. An example that uses quadtree subdivision with merging is illustrated in figure 3.6. The inaccuracy of borders and need for merging are quite apparent with quadtree subdivision, and the work of [31] suggests that for range image segmentation a Delaunay tessellation is more suitable.

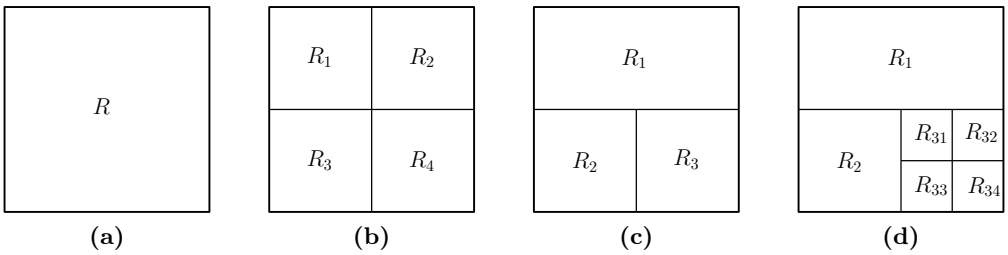


Figure 3.6: An example of a split-and-merge algorithm that subdivides regions into quarters: (a) first the homogeneity of the image as a whole is considered, (b) then it is subdivided into quarters, (c) this is followed by a merging step that proceeds to combine two regions that are similar to each other, and (d) splitting continues for regions that are not sufficiently homogeneous.

The results of region splitting approaches using quadtree subdivision and Delaunay tessellation are shown in figure 3.7. In this example the quadtree approach uses a similarity constraint where a plane is fit to each region using least squares and the region is subdivided if the residuals per pixel in the region are higher than some threshold. With the Delaunay approach a triangulation is created for the entire image from some set of points and each triangle is a planar region. The similarity constraint for each region is that the point furthest from the plane is closer than some threshold value. The method of region subdivision is then to include that furthest point in the Delaunay triangulation process which introduces new triangles that fit closer to the data.

A major computational disadvantage of region splitting is that any measure of homogeneity needs to consider large regions at a time, which is a costly operation and also one which often needs to be performed multiple times on some parts of the image. Another disadvantage is the extreme inaccuracy around region boundaries because of the subdivision process, which is either handled with a sensitive homogeneity measure that creates many small regions that subsequently need to be merged, or with a post-processing step that reconsiders region boundaries. It is therefore often necessary to use region growing to correct some of the flaws introduced during region splitting, and approaches that start with an over-segmentation (usually simply the image pixels) and then concentrate solely on region growing are more common than region splitting.

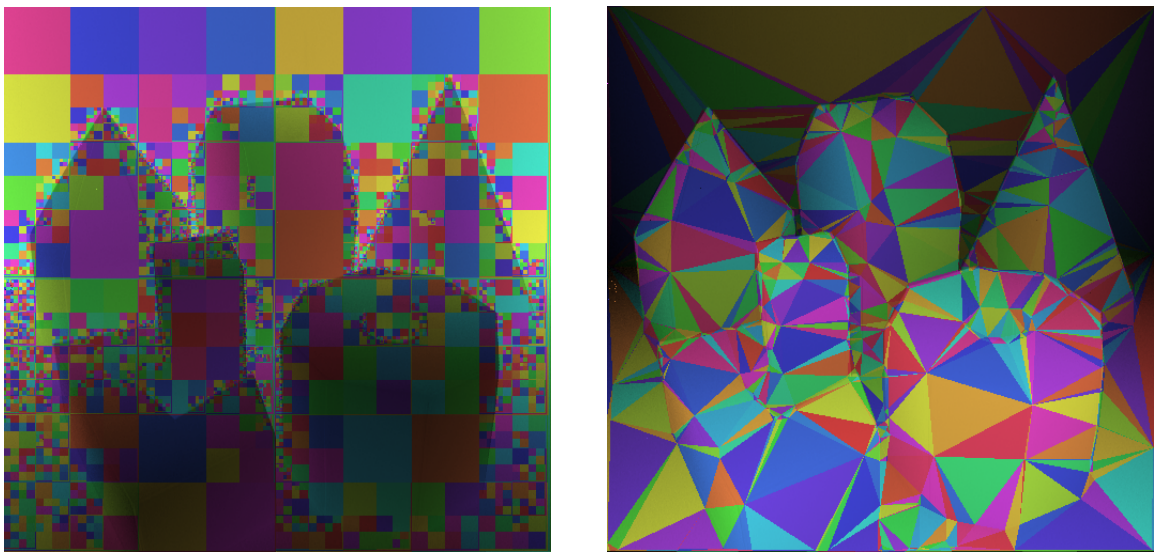
3.2.2 Region growing

Region growing can be considered the opposite of region splitting and is a bottom-up segmentation process that proceeds by merging regions to progressively grow larger. It might require that the image is first divided into small homogeneous regions, although the pixel structure of the image is often used for this purpose. The method then generally proceeds as follows.

1. Start a new region by selecting a seed region/pixel that has not yet been selected or merged (if no seed regions/pixels remain, proceed to step 4).
2. Merge adjacent regions/pixels that are considered sufficiently similar to the current region.
3. If no adjacent regions are sufficiently similar, stop growing and proceed to step 1, otherwise continue growing by repeating step 2.
4. All remaining regions should be homogeneous, with separate regions differing from one another.

This approach can be implemented in two ways, called iterative and simultaneous region growing. Iterative region growing proceeds to grow each region sequentially and suffers from biasing regions depending on the order in which they are processed. Simultaneous region growing is more parallelizable as it grows multiple regions at the same time, which offsets the bias caused by the iterative process, although it requires added complexity to handle region conflicts in an appropriate manner. When compared to region splitting, region growing has computational advantages because its bottom-up approach means that certain areas of an image rarely need to be considered more than once. Also, as long as the initial division of regions is small enough (e.g. pixel-sized), it theoretically allows all possible shapes consisting of valid connected regions, although in practice shapes that consist of large regions that are connected through thin regions are quite likely to be separated by noise, and smaller regions can easily be missed.

The biggest disadvantage of region growing is thus its dependence on the choice and possibly the ordering of the initial seed pixels. Various methods for selecting seed pixels have been developed.



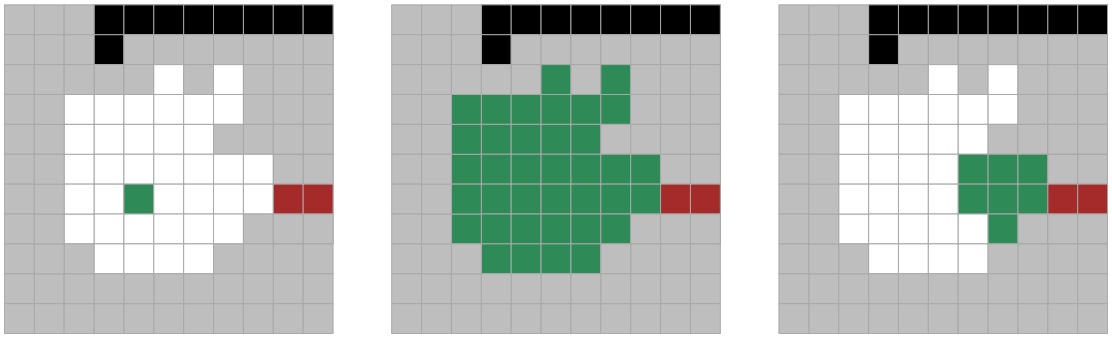
(a) quadtree **(b)** Delaunay triangulation

Figure 3.7: Region splitting approaches, such as (a) quadtree subdivision that splits regions into equal-sized quadrants and (b) Delaunay triangulation that takes advantage of the point cloud structure, have problems with handling object boundaries. To maintain accuracy the splitting process is often performed until definite over-segmentation, so that regions can subsequently be reformed through a merging process.

These methods generally focus on finding locations that are far from areas where the feature space is noisy so that they are in an area that can clearly be distinguished as a separate region. This assures that the region growing process generates regions that can be better trusted and allows the handling of more complex noisy areas to be left for later once more is known about the overall structure of the regions in the image. Seed points can be chosen by selecting points in the image with lowest curvature or by using edge detection methods and selecting points that lie as far away from any edges as possible.

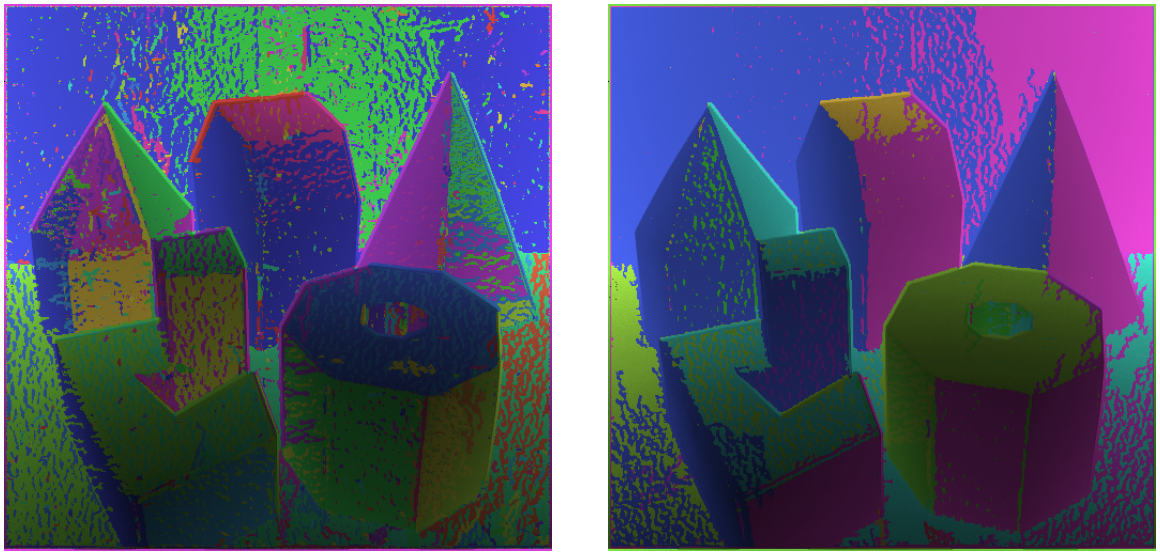
When determining how to expand, region growing considers all possible neighbouring regions and evaluates whether they are sufficiently similar to be merged with the current region. Generally, the first step is to find the pixels adjacent to the borders, usually with a morphological dilation. The regions these pixels belong to are then evaluated for similarity to the current region as a whole, to the properties of the initial seed point of the current region, or to only the properties of the current region that are near the region being considered. Each of these approaches is illustrated in figure 3.8. The choice of approach depends on the feature space used and how it changes across regions. Comparing with only the seed pixel leads to regions that are more strictly homogeneous, but the over-dependence on a correctly chosen seed pixel makes it highly susceptible to noise. Comparing against the average properties of the current region is more robust to noise and gains stability as the region grows larger and merged regions have less influence on the average properties of the region. Comparing against some part of the current region that is close to the region being considered for merging allows the region to handle a larger variety of different features, as long as the change between them is gradual. This is more useful for non-planar segmenters that consider curved regions where normals would change gradually. Allowing gradual changes is also dangerous though, as it means that merging pixels can be biased by the direction of approach, which adds another over-dependence on the initial selection of seed pixels.

Each decision to grow a region often requires multiple parameters that are used as thresholds for determining whether the pixels to be added are valid. Sometimes an intricate set of parameters is required to avoid cases where fewer parameters cannot deliver satisfactory results, as can be seen with example outputs shown in figure 3.9. The choices and parameters used for growing regions can vary extensively between approaches.



(a) seed pixel **(b)** whole region **(c)** nearby parts of region

Figure 3.8: Different parts of a region can be used when comparing to a new region being considered for merging. Grey pixels indicate the possible region of expansion, a red pixel is the current region being considered for merging, and the green pixels are those taken into consideration when similarity is evaluated.



(a) high threshold **(b)** low threshold

Figure 3.9: Region growing that depends on a single unchanging threshold often cannot properly segment the varying types of detail in an image. While a high threshold generally causes over-segmentation and a low threshold generally leads to under-segmentation, both types of errors can still occur at the same time, causing no threshold to deliver adequate results.

3.2.3 Robust estimator segmentation

Robust estimators are statistical methods for fitting models to data. These methods are particularly resistant to data containing significant amounts of outliers. They have been developed in response to the flaws observed in least squares and other approaches where even obvious outliers can heavily skew the models fitted. Robust estimators are often compared in terms of their breakdown point which is their tolerance to the percentage of outliers the data can contain. Analytically determining the breakdown point is generally not possible [32] and most methods just provide an estimate from experimentation and comparison. The most important considerations are usually whether the breakdown point is more or less than 50%, that it is as high as possible, and what factors of the model or data would cause it to be lower.

The most common robust estimators are random sample consensus (RANSAC) and the Hough transform. Since many robust estimators follow an approach similar to that of RANSAC, its process will be briefly explained. As noted in [32], most robust estimators, including least median squares (LMedS), residual consensus (RESC), adaptive least k-th squares (ALKS), and maximum density power estimator (MDPE), follow a similar four-step approach as used by RANSAC. These steps consist of randomly sampling points, using them to generate a candidate model, evaluating the quality of the candidate, and then providing the parameters for the model with the best quality measure. This can also be seen as a hypothesis generation and verification approach [33]. In the case where the data is a point cloud that contains a flat plane and a large number of outliers, this four step process would proceed as follows.

1. Since a plane has three degrees of freedom, three points are randomly sampled out of all the data.
2. A plane is then fitted to these points.
3. The plane's quality is measured by the number of inliers. Inliers are identified as in the data that either lie directly on the plane, or lie close enough to it.
4. This sampling process is then performed numerous times to increase the probability of

finding a sampling of points that are all inliers on the plane. Once the process is complete, the parameters of the plane with the most inliers found are then parameters of the model.

The robustness is therefore derived by not relying on all the data or even on the majority of the data, but rather on the largest portion that consistently agrees with a set of model parameters being fitted.

When using robust estimators as the basis of a range image segmentation algorithm, the following approach is usually followed.

1. Use a robust estimator to find a best model-fit across (all, or some of the) unlabelled points in the image, and determine the model's parameters.
2. Find all unlabelled points that are inliers for the parameters found.
3. The largest region of connected inliers is defined as a new region, and all remaining points are left unlabelled.
4. If the number of remaining unlabelled points is too few, or the quality of model-fit parameters found is too low, stop segmentation and identify remaining unlabelled points as noise. Otherwise, proceed to step 1.

An example is shown in figure 3.10 where RANSAC is used to find planes. In practice, these approaches often only look at a subset of the whole image or else use a hierarchical sampling technique that considers the image at different scales as in [32] and [34]. This is because regions that are a small part of the whole can often occur, and would otherwise not be found due to the breakdown limit of most robust estimators. For this type of approach, robust estimators are required that can handle outliers, pseudo-outliers, noisy inliers and possibly even clustered outliers. Pseudo-outliers are outliers that represent valid data, which are merely extraneous to the single model being fit. The robust estimators that have been used specifically for range image segmentation include least median squares (LMedS) [35], residual consensus (RESC) [36], adaptive least k-th squares (ALKS) [37], mutual inlier ratio (MIR) [38] (used in a region growing approach), adaptive scale sample consensus (ASSC) [34], maximum density power estimator (MDPE) [32], m-estimator sample consensus [39] and an estimator based on genetic algorithms [40].

As can be seen, there are many different robust estimators available and even more that have not been mentioned, such as least trimmed squares (LTS), iteratively reweighted least squares (IRLS), minimum unbiased scale estimator (MUSE), MLESAC [33] and MINPRAN. More detail about some estimators, including comparisons, can be found in [32] and [34].

3.3 Global optimization

Optimization approaches can be used to solve a wide variety of problems. Any problem whose solution can be evaluated with an objective function can be analysed with an optimization approach. The objective function can also be called an energy function and the goal of optimization is to find the configuration of input values that find either the minimum or maximum output value of this function. Since the choice of minimization or maximization is rather arbitrary (any energy function is easily adapted to either approach), the process will consistently be referred to as energy minimization and the energy function will be defined as follows:

$$y = E(x_1, x_2, ..., x_n). \tag{3.8}$$

This function takes multiple input variables, x_1 to x_n that define the solution being evaluated, and then provides a single output value y, which describes the quality of the solution. A lower

y means the solution is more preferable. The goal is then to define and traverse the space of possible input parameters so that the most suitable solution can be found. The energy function and even the amount of input variables do not need to be constant during optimization, and can be adapted for each solution being evaluated. Because of this, the parameter space of possible input variables will be referred to as f, and the goal of energy minimization will be defined as finding

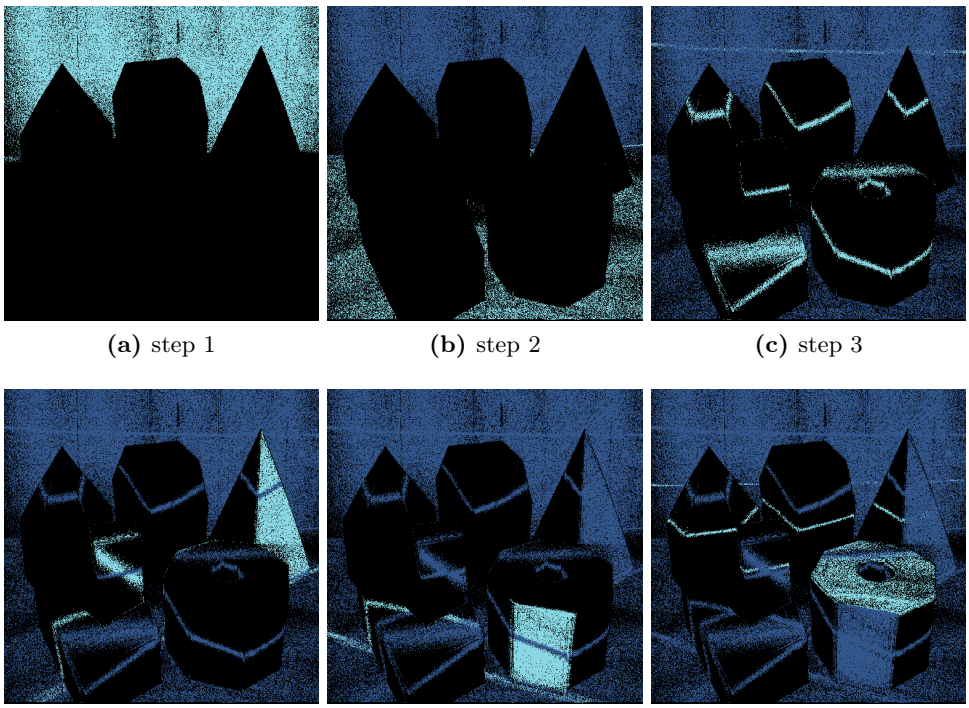
$$f_{\min} = \arg\min_{f} E(f). \tag{3.9}$$

The formulation of a problem in terms of an optimization approach then requires two parts:

- defining an energy function whose minimum output occurs with the input parameters that provide the most suitable solution;
- choosing an optimization approach that is likely to find a solution close to the optimal minimum (adjusting the energy function to make this possible, might be required).

If the problem being solved is well understood, then defining an energy function is often a relatively intuitive task as one need only consider which desirable properties an applicable solution should have. There are often various possible choices, each causing an energy function with a different solution space. Unfortunately the shape of the energy function is so deeply intertwined with the viability of finding a minimum that the choice of optimization method cannot be divorced from the development of the energy function.

As the possible parameter spaces and energy functions are so variable, there are an enormous variety of approaches for optimization. If the parameter space is discrete, then combinatorial



(a) step 1 **(b)** step 2 **(c)** step 3

(d) step 4 **(e)** step 5 **(f)** step 6

Figure 3.10: Progressive steps of a robust estimator segmentation approach are shown. Using a robust estimator, each step is likely to find one of the largest planes in the current scene (shown as light blue points). For the next steps all the points that have already been assigned to regions are removed from consideration (shown as dark blue points) so that the next largest plane can be found. In this case, an error has been made in step 3 where points from various planes were considered to be inliers of a fictional plane running across the image.

optimization approaches can be used, and if the solution space is small enough, an exhaustive search might even be possible. In general though, exhaustive searches are highly impractical and this means that optimization methods need to traverse some subset of the solution space in manners that improve the probability that the solution found is actually optimal.

For instance, if the energy function has a convex shape, then it can be optimized by selecting any starting point in the solution space, determining the local gradient at that point and then using the direction and magnitude of greatest gradient descent to select the next point. This approach is called gradient descent optimization and generally finds the minimum with a relatively small number of evaluations. Unfortunately, when this same approach is used on energy functions containing multiple local minima, it will find a local minimum unless its starting location is chosen carefully (see figure 3.11). Local optimization approaches are characterized by the property that they can get caught in a local minimum that is not the global minimum within the whole parameter space, although local optimizers can often be used as global optimizers for a specific class of energy functions. Local minimizers are thus generally quite useful and many have been used to develop approaches that have proven successful.

Ideal energy functions are those with few local minima, although any energy function whose local minima are well enough understood so that the number or location(s) of local minima can be predicted, can often be optimized with a specialized optimization approach. Approaches that can be used on any general energy function are not impossible, and have in fact been quite well researched. These general energy optimizers include methods such as simulated annealing, particle swarm optimization and genetic algorithms, some of which will be discussed in more detail. Of course, whether a method finds a global or local optimum is also determined by the energy function being optimized, and possibly the way the method is implemented, meaning that so-called global optimization methods do not always fit that definition in practice. General global optimization approaches are only guaranteed to find the global optimum if they are allowed to search for an indefinite amount of time, and methods that are more specialized to the specific energy function being minimized are often faster and provide better solutions.

A significant disadvantage of optimization approaches is that they are generally slow and computationally expensive as they have to analyse the energy function an enormous number of times. Also, when developing an optimization approach, both the energy function and optimization methods need to be kept in consideration. When results are poorer than expected it is then hard to know whether the fault lies in the energy function whose minimum does not provide the desired result, or whether the optimization method merely provided an answer that is not close to the global minimum. This is even more problematic with general optimization approaches that do not provide any strong assurances that a global minimum will be found.

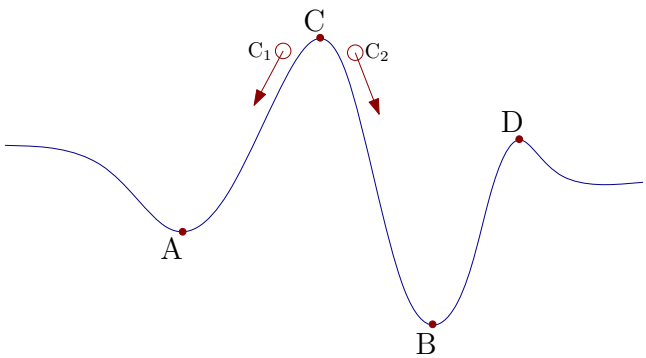


Figure 3.11: A function with a local minimum (A), a global minimum (B), a global maximum (C) and a local maximum (D). If locations C_1 and C_2 are used as starting locations for a gradient descent approach, they will respectively lead to two different local minima (A and B), only one of which is the global minimum (B).

On the other hand, a notable advantage is that the energy function can present an intuitive understanding of why a solution was chosen and most parameters it contains are usually not arbitrary and often have a physically meaningful interpretation. Also, when the optimization method that is used works sufficiently well, it can almost be ignored and focus can be directed towards what type of solution is required, instead of how that solution needs to be found. Most segmentation approaches discussed so far cannot guarantee any type of optimality, and approaches that only look at some subset of the image at a time (or some single property calculated from the whole image) are, in theory, unlikely to find anything but sub-optimal results.

With range image segmentation the parameters of the energy function are often the parameters of models being fit to the data, and the energy function determines the a priori likelihood of such a configuration of models occurring, as well as the likelihood of those models being correct considering the observed data. In image processing there are many possible global energy optimization approaches, mostly based on the concept of formulating the problem as a Markov random field and then maximizing the joint distribution [41]. These include simulated annealing, mean field annealing, graduated nonconvexity, graph cuts and genetic algorithms. This study only considers simulated annealing which has already been used for range image segmentation, and graph cut optimization from which a segmentation approach is developed.

3.3.1 Simulated annealing

Simulated annealing is a general optimization approach that is based on the physical process of annealing. This physical process is to heat metals to higher temperatures and then allow them to cool slowly which results in a material containing fewer flaws. The reason for this is that while atoms prefer energetically lower states, they can get stuck in flawed configurations that are sub-optimal with higher energy but from which they do not have enough energy to escape and find a better configuration. Heating them then provides the atoms enough energy to break free and move to other configurations, while cooling forces them to stay in these new configurations. They are cooled slowly so that they still have time to move to different configurations before they lose too much energy and settle into the most stable low energy state that they can achieve.

In simulated annealing this process is analogously duplicated by introducing a 'temperature' variable that controls how much the parameters are randomly altered during the optimization process. This temperature variable is then changed according to a cooling schedule that steadily decreases the parameter variations made until the system reaches a stable low energy state. The variation in parameters is done in a probabilistic manner such that large changes become less likely with lower temperatures, but never impossible. Theoretically, this ensures that the process cannot get stuck in local minima, as random changes that would escape the local minimum are always still possible.

In the work of Han et al. [42] simulated annealing is used to optimize an energy function designed to segment range images into various surface models. Their framework is notable for being general enough to allow various types of surface models to be used, as can be seen in figure 3.12. Their energy function is based on maximizing the Bayesian posterior probability over their solution space, Ω. When considering possible solutions $W \in \Omega$, they search for

$$W^* = \arg\max_{W \in \Omega} p(W|D, I) = \arg\max_{W \in \Omega} p(D, I|W)p(W) \tag{3.10}$$

where D is the depth data and I is intensity data which, corresponding to the laser range capturing approach, describes the strength of the reflected signal. The likelihood $p(D, I|W)$ is determined by the distance of the depth data from the surface models, as well how well the data fits with the reflectance model (the distance data is also partially weighted by the intensity values as they also indicate the dispersion of the laser signal). The prior $p(W)$ of a solution W

is determined by a combination of factors, including the number of surface parameters needed to describe W, the size of the surfaces, the smoothness of surface boundaries, a prior such that any spline models are as planar as possible and a prior that assumes the angles at a corner are more or less equal. They then have only a single parameter α which is a scale factor controlling the scale of segmentation and which is necessary in the prior on surface sizes. Unfortunately this generality also means that their solution space is huge and high dimensional, and they require advanced techniques for traversal of this space. To improve the efficiency of traversing the solution space, a number of data-driven techniques are incorporated that attempt to reduce the solution space by removing possibilities that are unlikely according to the data. Unfortunately this also requires delicate adjustments to prevent these improvements from getting the optimizer stuck in a local minimum.

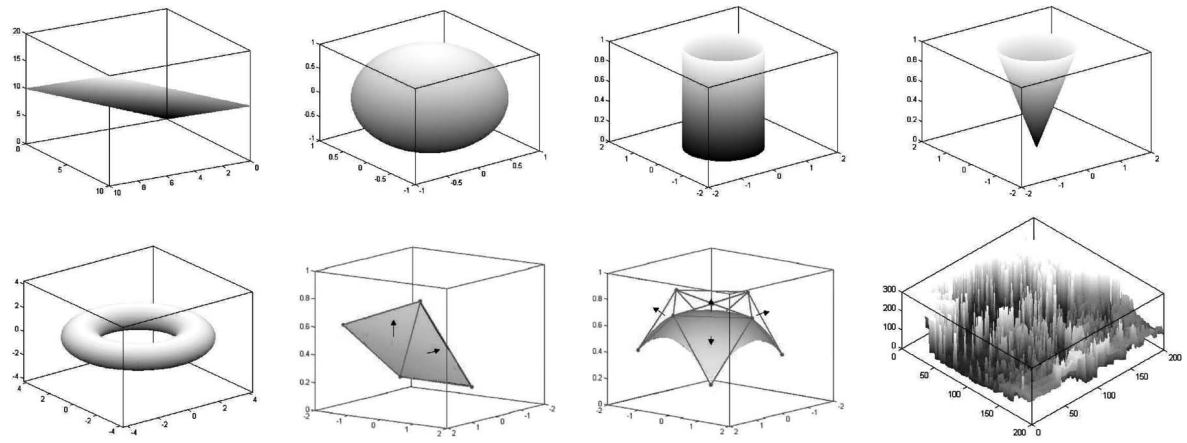


Figure 3.12: To segment real-world ranged scenes, [42] uses five families of surfaces as models to fit to the range data. These surfaces present typical examples from these families and include a plane, sphere/ellipsoid, cylinder, cone, torus, 4-point spline, 9-point spline and clutter. The sphere, cylinder, cone and torus are all modelled using a single conic surface model that uses seven parameters and the clutter is modelled with a non-parametric 3-D histogram. Image source: [42].

3.3.2 Graph cut optimization

Graph cut optimization works by setting up a graph in such a manner that the minimum cut on the graph is also a solution that minimizes a specific type of energy function. Graph cut optimization is thus an optimizer that is much less general than simulated annealing. A specific graph structure has to be designed according to the energy function being optimized and, while the energy functions for which this is possible are limited, it has been shown that there are large classes of applicable functions and methods have even been developed to automate the construction of the necessary graphs once a function has been chosen [43].

This approach has found many applications in computer vision such as image restoration, image segmentation and multi-camera scene reconstruction [43]. What makes this approach so useful is that there are often strong guarantees about the quality of the solution, often that it is provably a global optimum. The guarantee of finding a global optimum is of great value as it is then no longer necessary to design both the energy function and the optimization approach at the same time. It then only becomes important whether the desired solution to the problem being found can be fitted into an applicable energy function. Unfortunately, these guarantees are frequently only for functions whose inputs consist of binary variables.

The work of Veksler [44] shows a specific energy function that is general enough to handle

many applications in computer vision as it is associated with a probabilistic interpretation of the observed values of an image. Furthermore, it is then proven that a graph cut formulation of this energy function provides a solution that is provably close to the global optimum, even for the more complex multi-label case of the problem.

While graph cut optimization has been used for many problems relating to colour images, a study of the literature shows that it has not yet been used to attempt segmentation of range images, with the closest approach being the interactive segmentation of point cloud data [45] and also approaches that perform multi-view reconstructions of point clouds [46, 47, 48]. For these reasons and the advantages listed above, graph cut optimization will be investigated to consider how it works and how it can be applied to the problem of the planar segmentation of range images.

Chapter 4

Multi-label graph cut optimization

We will attempt to develop a segmentation method based on multi-label graph cut optimization, which is why the theory behind graph cut optimization will be considered in more depth so that its capabilities and limitations can be better understood.

This chapter describes some of the theory for multi-label graph cut optimization as defined and developed in the work by Veksler, Boykov, Zabih and Kolmogorov [44, 49, 50]. The labelling problem is first explained as it provides a common framework and notation for describing many diverse problems.

4.1 The labelling problem

Many problems in computer vision can be posed as a labelling problem in which the solution is a set of labels assigned to image pixels or features. For a discrete set of sites $\mathcal{S} = \{1, 2, \ldots, n\}$ and a discrete set of labels* $\mathcal{L} = \{\ell_1, \ldots, \ell_k\}$, the labelling problem is that of assigning a label from $\mathcal{L}$ to every site in the set $\mathcal{S}$. This mapping from $\mathcal{S}$ to $\mathcal{L}$ is called a labelling, or configuration, and is denoted by $f = \{f_1, \ldots, f_n\}$ where $f_p \in \mathcal{L}$ for every $p \in \mathcal{S}$. The set of all labellings is denoted by $\mathcal{F}$ and consists of k^n possible configurations.

An energy function $E(f)$ can be used to evaluate a particular labelling f, such that the problem of finding some optimal labelling then becomes one of minimizing the energy function over the solution space $\mathcal{F}$.

4.2 Energy functions in vision

Energy functions of the form

$$E(f) = E_{data}(f) + \lambda \cdot E_{prior}(f) \tag{4.1}$$

are popular for computer vision problems. The $E_{data}(f)$ term encodes information about how much a configuration deviates from the observed data, and the $E_{prior}(f)$ term compares the configuration against prior assumptions about the dependency between sites. The constant λ allows for the adjustment of relative importance carried by each term.

*The labelling problem also allows the labelling set to be continuous but, as will be shown later by the graph structures that are used, this is not the case for solutions provided by the graph cuts formulation.

Due to the nature of $\mathcal{S}$ and $\mathcal{L}$, the search space is typically extremely large and high dimensional, and finding the global minimum of E is a significant problem for such a general formula. For many formulations of this energy function the problem is intractable and NP-hard, but there are useful classes of energy functions that can be minimized efficiently using a graph cuts optimization approach.

4.3 Energy functions for graph cuts

The multi-label graph cut optimization approach described in the rest of this chapter assumes an energy function where the prior knowledge term $E_{prior}(f)$ favours a smoothness between neighbouring sites and where the data term $E_{data}(f)$ assumes observational independence. The formalization of these constraints is presented here, and is based on the dissertation of Veksler [44].

The energy term for the data should penalize labelling configurations that do not agree with the observed data values. While the actual choice of energy values is highly dependent on the choice of what the labels represent, the noise in the data and the overall energy function, a more specific data term can still be defined as

$$E_{data}(f) = \sum_{p \in \mathcal{S}} D_p(f_p). \tag{4.2}$$

Here $D_p(f_p)$ is defined as a measure for how much assigning the label f_p to the site p disagrees with the observed data. This definition assumes that the observed value of each site is statistically independent and the only restriction on $D_p(f_p)$ is that it cannot be negative.

The energy term for describing the prior assumptions about the structure of the scene first requires a definition of a neighbourhood structure over the sites. For a site p, a set of sites N_p, called its neighbourhood, is defined and each site in the set N_p is called a neighbour of site p. For any two sites p and q it is required that the neighbourhood structure satisfies the following properties:

- $p \notin N_p$;
- if $p \in N_q$ then $q \in N_p$.

This means that the relationships between sites can be represented with an undirected graph. The only further constraint on this neighbourhood graph is that it cannot contain cliques with more than two vertices*, although this still allows the representation of many different spatial relationships. A clique is defined as a set of sites where each member of the set is a neighbour of all other members. In computer vision the most commonly used neighbourhood structure is the 4-connectivity neighbourhood structure shown in figure 4.1. Its grid pattern neatly describes the relationships between pixels in an image, and any method of grouping pixels into regions can still use a related neighbourhood structure.

If a neighbourhood system $\mathcal{N}$ is defined as the set of all neighbouring pairs of sites $\{p, q\}$, then a smoothness prior can be defined as

$$E_{prior}(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p, f_q), \tag{4.3}$$

where $V_{\{p,q\}}(f_p, f_q)$ is a neighbour interaction function that penalizes neighbouring labels that are different. The penalty depends on the labels f_p and f_q assigned to the pixels p and q, and

*In [51] work is done that extends graph cut optimization to work with graph structures that contain higher-order cliques.

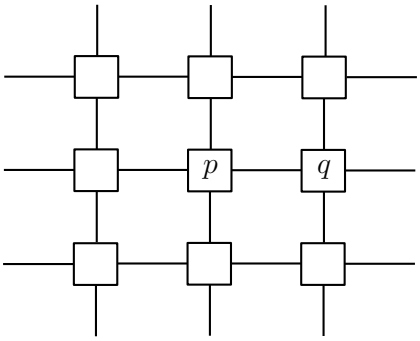


Figure 4.1: A graph showing a 4-connectivity grid neighbourhood structure. Note that this structure does not have cliques of size larger than 2.

might also differ for every particular pair of neighbours p and q. By looking only at pairwise interactions, we are limited to only being able to model the first derivative of a labelling and no higher order derivatives. The limitation on $V_{\{p,q\}}$ is that the function has one of a few specific forms, called smoothness priors, for which specific optimization properties have been determined and can be guaranteed. These smoothness priors and their optimization properties are discussed in more detail in section 4.5. The smoothness energy term is thus simply the sum of all the penalties as determined by the neighbour interaction functions for all neighbouring pairs of sites.

The general form of the energy function to be minimized is therefore

$$E(f) = \sum_{p \in \mathcal{S}} D_p(f_p) + \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p, f_q), \tag{4.4}$$

which agrees with the terms in equation (4.1) as long as λ or its inverse is incorporated into either $V_{\{p,q\}}(f_p, f_q)$ or $D_p(f_p)$. While this study calls the two terms the data and smoothness terms, they are also often referred to as the region and boundary terms or as the unary and pairwise terms. This is because they respectively describe the penalty function between a single site and its observed value, and the interaction function between pairs of sites which are chosen as a smoothness constraint. The approach necessary to define these terms, and motivate minimizing this specific energy function, can be found from a probabilistic context by describing the labelling problem in terms of maximizing the joint distribution of a Markov random field.

4.4 MAP-MRF framework

Minimizing equation (4.4) can be motivated by its equivalence to maximizing the joint distribution of a Markov random field through maximum a posteriori estimation [44]. The required Markov random field and its joint distribution are described first, and it is then followed by an explanation of the maximum a posteriori estimation.

4.4.1 Markov random field joint distribution

In the labelling problem, each site can be considered to correspond to a random variable whose distribution function is dependent on the observed values at that, and all other, sites. If it is assumed that noise in the observed values is statistically independent from one another then the random variable of a single site need depend only on its own observed value (or values), and the

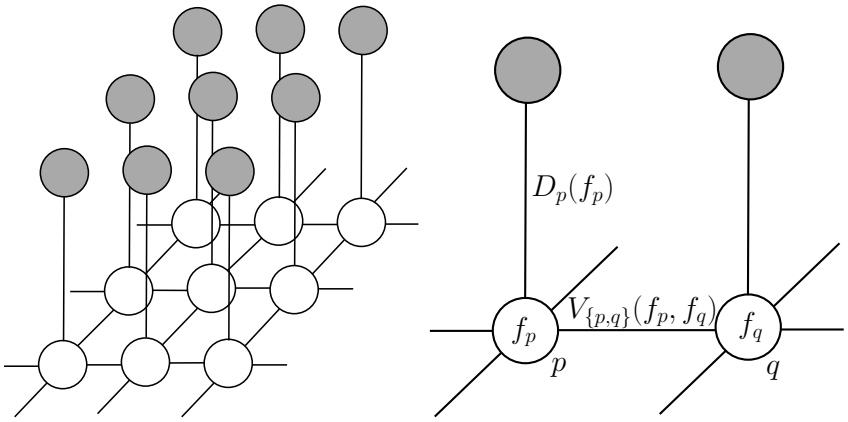


Figure 4.2: Markov random field representing the possible labels of a 4-connected neighbourhood of sites and their relationship with the observed values.

relationships between all the sites can be fully described by a Markov random field*.

A Markov random field (MRF), which is also called an undirected graphical model or Markov network, is very useful in modelling spatial interactions and relationships between various sites and also their observed values. Once these relationships are described in an MRF, the goal is to consider all the relationships and find the most likely configuration of values (or labels) to assign to the sites. This is done by considering the joint probability distribution function over the MRF.

If the neighbourhood structure is a 4-connected neighbourhood, then the MRF that will be considered is of the form shown in figure 4.2. The grey vertices represent the observed values and all the edge weights use the functions from the energy function in equation (4.4), which will be further described here.

As long as the probability distribution function is positive for all configurations, the Hammersley-Clifford theorem [52] states that any joint distribution function of a Markov random field is equivalent to the Gibbs distribution† that can be fully specified by looking only at the cliques of the network. The Gibbs distribution is

$$P(f) = Z^{-1} \cdot \exp\left(-\sum_{c \in \mathcal{C}} V_c(f)\right), \tag{4.5}$$

where Z is a normalizing constant, $\mathcal{C}$ is the set of all cliques, and $V_c(f)$ are called the clique potential functions that map a labelling to a real number. Therefore only these clique potential functions are required to specify the MRF.

These potential functions are then defined to be zero for all cliques of size larger than two and those of size two have the potential function

$$V_c(f) = V_{\{p,q\}}(f_p, f_q) \tag{4.6}$$

as defined in section 4.3. The joint distribution function of the MRF then becomes

$$P(f) = Z^{-1} \cdot \exp\left(-\sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p, f_q)\right). \tag{4.7}$$

*If statistical independence of the observations is not assumed, then a conditional random field is necessary. A conditional random field is also necessary if the neighbourhood interaction between pairwise sites is dependent on the observations.

†Also called the Boltzmann distribution and generalized as the Gibbs measure.

4.4.2 Maximum a posteriori estimation

An observation d is related to the unknown field configuration f by the likelihood function $P(d|f)$. The most popular method of estimating the field configuration of an MRF is by the maximum a posteriori estimation (MAP) which consists of maximizing the posterior probability. Using Bayes' theorem the posterior probability can be written as

$$P(f|d) = \frac{P(d|f)P(f)}{P(d)}. \tag{4.8}$$

The MAP estimate would therefore be equal to

$$f^* = \arg\max_{f\in\mathcal{F}} P(d|f)P(f), \tag{4.9}$$

since $P(d)$ is not dependent on f. If d_p is defined to be the observation at pixel p, and it is assumed that the noise at each pixel is independent, then $P(d|f)$ can be modelled as

$$P(d|f) = \prod_{p\in\mathcal{S}} P(d_p|f_p). \tag{4.10}$$

If it is then further assumed that

$$P(d_p|l) = C_p \cdot \exp\left(-D_p(l)\right), \quad \text{for } l \in \mathcal{L}, \tag{4.11}$$

where C_p is the normalizing constant, and D_p is defined as in section 4.3, then the likelihood can be written as

$$P(d|f) \propto \exp\left(-\sum_{p\in\mathcal{S}} D_p(f_p)\right). \tag{4.12}$$

With the above definitions for $P(d|f)$ and $P(f)$, a MAP estimate can now be written as

$$f^* = \arg\max_{f\in\mathcal{F}} \exp\left(-\sum_{p\in\mathcal{S}} D_p(f_p) - \sum_{\{p,q\}\in\mathcal{N}} V_{\{p,q\}}(f_p, f_q)\right). \tag{4.13}$$

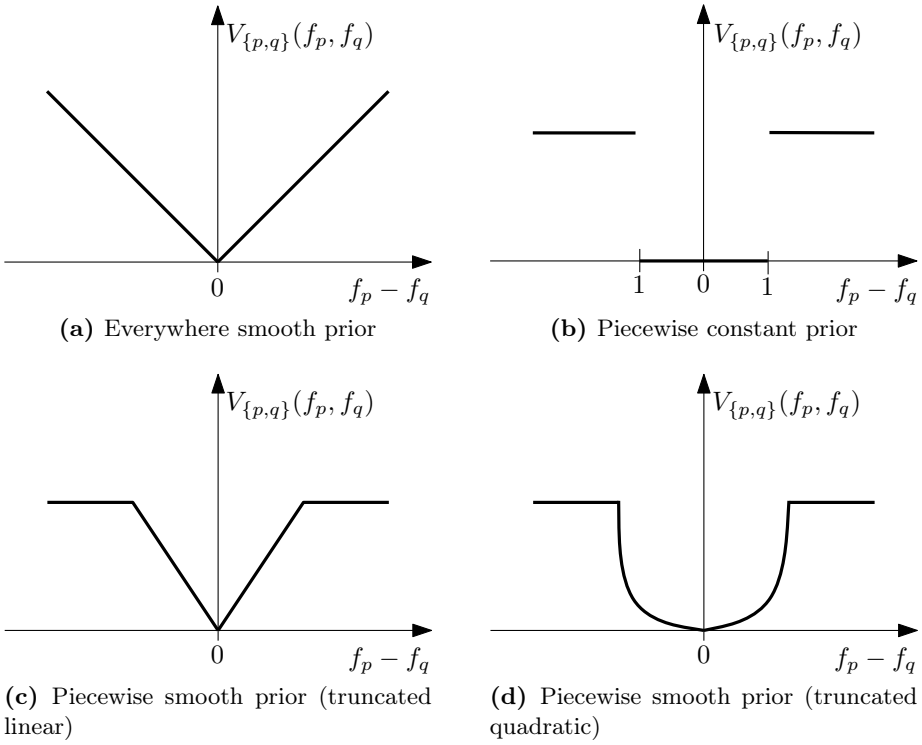
This shows that maximizing the joint probability over this MRF is equivalent to minimizing

$$E(f) = \sum_{p\in\mathcal{S}} D_p(f_p) + \sum_{\{p,q\}\in\mathcal{N}} V_{\{p,q\}}(f_p, f_q), \tag{4.14}$$

which is identical to equation (4.4). Now that there is sufficient reason to look at minimizing this type of energy function, it will be discussed how graph cuts can be used to perform this task. Before explaining graph cuts, the smoothness term defined in equation (4.3) will be considered in greater detail. The choice of function for the smoothness term determines which type of graph structures can be used.

4.5 Smoothness priors

The smoothness term in equation (4.3) assigns a penalty for labelling configurations that contain abrupt changes and do not fit the assumption that the data is inherently smooth. This assumption is made prior to observing any data and the functions used for this penalty cost are thus called smoothness priors. Three priors, namely the everywhere smooth, piecewise smooth and piecewise constant priors (which are illustrated in figure 4.3), are considered.



(a) Everywhere smooth prior

(b) Piecewise constant prior

(c) Piecewise smooth prior (truncated linear)

(d) Piecewise smooth prior (truncated quadratic)

Figure 4.3: Some possible smoothness priors. Priors (a) to (c) are all metric and the piecewise smooth prior is the only one here that can include semi-metric functions such as (d).

4.5.1 Everywhere smooth prior

The everywhere smooth prior assigns a low cost to labellings that are smooth everywhere. Labels that change gradually across an area are thus penalized much less than labels that contain abrupt changes and discontinuities. This is done by penalizing neighbouring pixels proportional to the difference in their labels such that larger differences give higher penalties. One example of a function that has these attributes is $|f_p - f_q|$, illustrated in 4.3(a), where the absolute difference between labels of two neighbouring pixels is used. For the work in [44], a graph structure is shown that can be used to globally optimize a similar function which includes an additional scaling factor,

$$V_{\{p,q\}}(f_p, f_q) = u_{\{p,q\}}|f_p - f_q|. \tag{4.15}$$

The scaling factor $u_{\{p,q\}}$ can be defined uniquely for each pair of pixels. This function has its limitations as it requires that all labels are in correspondence with a subset of integers in a meaningful manner and the only way to adapt it to a problem is by defining the order of the labels and the relevant scaling factors. The work of [44] shows that discontinuities in the data are not handled well when this prior is used, but that using a variable value for the scaling factor can significantly improve the results.

4.5.2 Piecewise smooth prior

The piecewise smooth prior assigns a low cost to labellings that change gradually, but does not overly penalize discontinuities, allowing them to be introduced when necessary. It is therefore more lenient towards labelling configurations that consist of smooth regions wherein any changes are gradual, but which are separated from one another by abrupt changes. This is done by limiting the penalties that are assigned to much larger changes. A similar function to the everywhere smooth prior can therefore be used, but it has to be non-decreasing in $|f_p - f_q|$ and bounded

from above. The bound prevents large changes from being overly penalized, which encourages discontinuities. Two examples are the truncated linear function, illustrated in figure 4.3(c),

$$V_{\{p,q\}}(f_p, f_q) = \begin{cases} u_{\{p,q\}}|f_p - f_q|, & \text{if } |f_p - f_q| < C, \\ u_{\{p,q\}}C, & \text{otherwise,} \end{cases} \tag{4.16}$$

and the truncated quadratic function, illustrated in figure 4.3(d),

$$V_{\{p,q\}}(f_p, f_q) = \begin{cases} u_{\{p,q\}}(f_p - f_q)^2, & \text{if } |f_p - f_q| < C, \\ u_{\{p,q\}}C^2, & \text{otherwise.} \end{cases} \tag{4.17}$$

In [44], methods are shown that can be used to optimize these piecewise smooth priors. These methods are general enough to be used for any metric or semi-metric functions. A semi-metric function satisfies the following properties:

$$\begin{aligned} V_{\{p,q\}}(l,l) &= 0, \\ V_{\{p,q\}}(l_1,l_2) &\geq 0, \\ V_{\{p,q\}}(l_1,l_2) &= V_{\{p,q\}}(l_2,l_1), \end{aligned} \tag{4.18}$$

while a metric function additionally satisfies the constraints

$$\begin{aligned} V_{\{p,q\}}(l_1,l_2) &> 0 \quad \text{if } l_1 \neq l_2, \\ V_{\{p,q\}}(l_1,l_2) + V_{\{p,q\}}(l_2,l_3) &\geq V_{\{p,q\}}(l_1,l_3). \end{aligned} \tag{4.19}$$

This latter constraint is a property called subadditivity which puts much stronger limitations on which functions can be metric. The functions in equations (4.15) and (4.16) are metric, while the one in equation (4.17) is semi-metric.

4.5.3 Piecewise constant prior

The piecewise constant prior is an important special case of the piecewise smooth prior. It is a metric function defined as

$$V_{\{p,q\}}(f_p, f_q) = u_{\{p,q\}} \cdot [1 - \delta(f_p, f_q)], \tag{4.20}$$

where

$$1 - \delta(f_p, f_q) = \begin{cases} 0, & \text{if } f_p = f_q, \\ 1, & \text{otherwise.} \end{cases} \tag{4.21}$$

The resulting energy function is

$$E_P(f) = \sum_{p \in \mathcal{S}} D_p(f_p) + \sum_{\{p,q\} \in \mathcal{N}} u_{\{p,q\}} \cdot [1 - \delta(f_p, f_q)]. \tag{4.22}$$

When the weight $u_{\{p,q\}}$ is constant the smoothness term is called the *Potts* energy as it comes from the Potts model. The Potts model is a generalization of the Ising model which is a mathematical model of the physical process of ferromagnetism which was a precursor to the study of Markov random fields [53].

4.6 Graph cuts

To understand how the energy function is minimized, it is first explained what a graph cut is. Then the algorithmic framework in which graph cuts are used is considered, as well as the

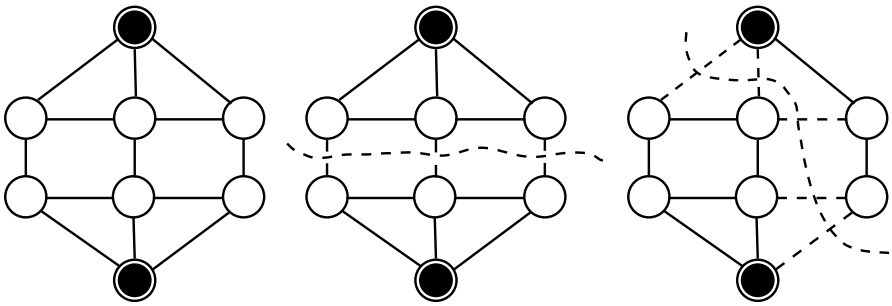


Figure 4.4: An undirected weighted graph with two terminal vertices, along with two possible graph cuts.

specific graph structures that are required such that the cost of a minimum cut corresponds to a minimum of a particular energy function.

A symmetric (undirected) weighted graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ consists of a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$. Each edge in $\mathcal{E}$ has a weight. Two vertices in $\mathcal{V}$ are defined as the terminals and are called the source and the sink. A cut through the graph is then defined as a set of edges $\mathcal{C} \subset \mathcal{E}$ such that the terminals are separated in the induced graph $\mathcal{G}(\mathcal{C}) = \langle \mathcal{V}, \mathcal{E} - \mathcal{C} \rangle$ and no proper subset of $\mathcal{C}$ separates the terminals in $\mathcal{G}(\mathcal{C})$. Two example graph cuts are shown in figure 4.4.

The cost of a cut $\mathcal{C}$ is denoted by $|\mathcal{C}|$ and is equal to the sum of its edge weights. The minimum cut problem is to find a cut with the smallest cost for a particular graph. According to a theorem by Ford and Fulkerson [54] this problem can be solved efficiently by calculating the maximum flow between the terminals, which is a problem for which many fast algorithms have been found. These algorithms have a low-order polynomial worst-case complexity which is dependent on the graph structure. For the methods developed in [44], it is mentioned that in practice the average complexity is nearly linear.

Here the graph structure over which the minimum cut must be found is determined by the specific energy function. For some energy functions a single graph can be set up such that its minimum cut corresponds to the exact global minimum solution. For other energy functions an iterative optimization approach is followed that traverses the solution space by finding the minimum cuts for various graphs that each find a local minimum within some move space.

4.7 Graph structures

An explanation is now given about how the graph structures are set up to optimize the everywhere smooth and piecewise smooth priors. While the everywhere smooth prior has a graph structure whose minimum cut corresponds to the global minimum for the multi-label case, the piecewise smooth prior does not, and uses a graph structure for the binary-labelling problem to find a minimum cut that corresponds to a local minimum in some chosen move space (discussed in section 4.8).

4.7.1 Everywhere smooth prior

For each pixel p a set of vertices $p_1, \ldots, p_{k-1}$ is created (k is the number of labels). These vertices are connected by k edges called t-links which correspond to the possible labels that can be assigned to the pixel p. The t-link edges are connected such that they form a path from source to sink. Thus, there are t-links between the source and p_1, between the sink and p_{k-1}, and between each p_{j-1} and p_j according to the consecutive order of the labels. Edges called n-links

are also created to connect each pair of neighbouring pixels p and q. Each of the n-links between p and q is assigned the weight $u_{\{p,q\}}$, while each of the t-links for each pixel p is assigned the weight $K_p + D_p(l_j)$ with $j \in \{1, \ldots, k-1\}$ and K_p any constant such that $K_p > (k-1)\sum_{q \in \mathcal{N}_p} u_{\{p,q\}}$. A subgraph of this graph, corresponding to two pixels p and q, is illustrated in figure 4.5.

Note that since a graph cut needs to separate the terminals, any valid graph cut needs to cut at least one or more of the t-links corresponding to a particular pixel, as they form a path between the terminals. Also note that the edge weights have been specifically set up such that the constant K_p is sufficiently large to force any minimum graph cut to only cut one t-link for each pixel. This can be seen by supposing that two t-links are cut for one pixel, in which case it is always possible to find a smaller cut by restoring one of the t-links and cutting all the n-links between p and q. The minimum cut will therefore always cut exactly one t-link for each pixel and, to determine the labelling configuration, the corresponding label for each cut t-link is then assigned as a label for each pixel. For a more detailed description of this graph structure, and proof that a minimum cut of it is equivalent to a global minimum for the corresponding energy function, the reader is referred to [44].

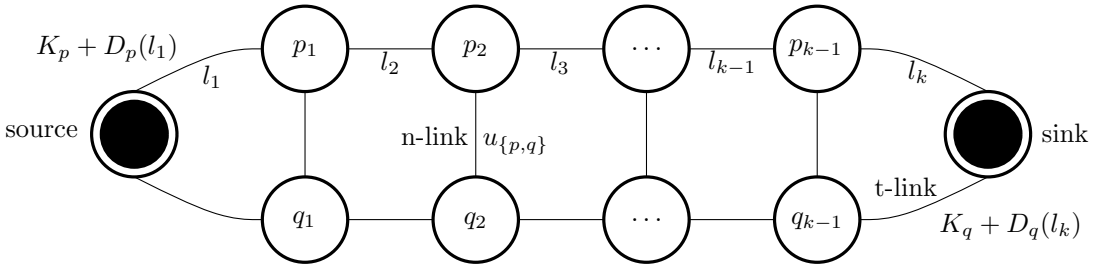


Figure 4.5: A subgraph corresponding to the pixels p and q for the graph $\mathcal{G}$ that is created to optimize an everywhere smooth prior with k labels.

4.7.2 Piecewise smooth prior

In the case of a piecewise smooth prior, it is not possible to set up a graph that can guarantee finding the global minimum for multiple labels, although it is possible to set it up to find a local minimum which, when performed multiple times in a manner determined by some specific move space, can be provably close to the global minimum.

To do this, a graph is set up to optimize for a labelling problem that considers only a subset of all the labels. The subset of labels is used to create a graph that exactly solves a binary labelling problem. The two terminals of the graph correspond to the possible labels that can be assigned to each pixel (unlike previously where the t-links corresponded to all the labels). In the case of an approach called α-β-swap (discussed in more detail in section 4.8) it is considered whether any two labels α and β should be swapped, and in the case of the α-expansion approach it is considered whether any pixels should be assigned the label α. Only the graph structure for the α-expansion approach is considered here. The one for α-β-swap works in a similar manner.

The graph structure for α-expansion is created dynamically according to the current labelling configuration f and some chosen label α. The graph structure incorporates all the pixels and their neighbours, but not for all possible labels, and additional auxiliary vertices are then added according to the current configuration f. An example is illustrated in figure 4.6. As can be seen in the example, a graph is set up with a set of vertices for each pixel in the image (in this case p, q, r, s). These vertices are directly connected to both terminals which are called α and $\bar{\alpha}$ and which correspond to the concept of labelling a particular pixel as α or keeping its label unchanged. The pixel vertices are then also connected to the vertices corresponding to their neighbouring pixels, either directly or indirectly through additional auxiliary vertices that

are added. These auxiliary vertices are placed between neighbouring pixels that currently have different labels, and they are then connected to the $\bar{\alpha}$ terminal. All the edge connections are then assigned weights according to the values listed in table 4.1. More details are in the work of [44], where it is shown that the minimum cut across this graph structure is equivalent to a local minimum in the α-expansion move space. It is then also proven that the combination of these local minimizations provides a result with certain guarantees of being close to a global minimum.

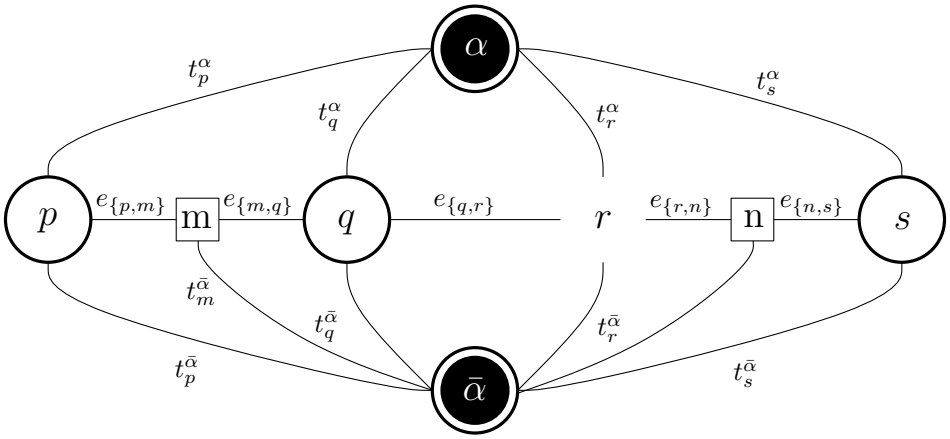


Figure 4.6: An example graph that could occur during an α-expansion move space optimization. In this case the image is 1-D with a set of pixels $\mathcal{S} = \{p, q, r, s\}$ and a neighbourhood structure $\mathcal{N} = \{\{p, q\}, \{q, r\}, \{r, s\}\}$. Auxiliary vertices m and n have been inserted according to the current hypothetical labelling configuration f where $f_p \neq f_q$ and $f_r \neq f_s$, while $f_q = f_r$ (and therefore no vertex is inserted between q and r). The specific edge weights are detailed in table 4.1.

Edge	Condition	Weight
t_p^α	$p \in \mathcal{S}$	$D_p(\alpha)$
$t_p^{\bar{\alpha}}$	$f_p = \alpha$	∞
$t_p^{\bar{\alpha}}$	$f_p \neq \alpha$	$D_p(f_p)$
$e_{\{p,a\}}$ $e_{\{a,q\}}$ $t_a^{\bar{\alpha}}$	$\{p, q\} \in \mathcal{N}, f_p \neq f_q$	$V_{\{p,q\}}(f_p, \alpha)$ $V_{\{p,q\}}(\alpha, f_q)$ $V_{\{p,q\}}(f_p, f_q)$
$e_{\{p,q\}}$	$\{p, q\} \in \mathcal{N}, f_p = f_q$	$V_{\{p,q\}}(f_p, \alpha)$

Table 4.1: This tables shows the edge weights used for the graph structures constructed during α-expansion. Source: [44].

4.8 Move spaces

The concept of traversing the solution space by looking for local minima can be neatly described by the concept of a move space. With the solution space of all possible labelling configurations as $\mathcal{F}$, there are then $|\mathcal{F}^2|$ possible ways to move from one configuration to another. Any set $\mathcal{M} \subset \mathcal{F} \times \mathcal{F}$ is a set of moves that will be called a move space. If $(f, f') \in \mathcal{M}$ then f is within one move from f'. A labelling f is then a local minimum within a move space $\mathcal{M}$ if $E(f) \leq E(f')$ for any $(f, f') \in \mathcal{M}$. If $\mathcal{M} = \mathcal{F} \times \mathcal{F}$ then a local minimum with respect to $\mathcal{M}$ is also a global minimum.

Local graph cut optimization approaches generally proceed in a manner similar to the iterative α-β-swap and α-expansion algorithms outlined in listing 4.1. Both these move spaces are examples of a category of optimizations that use graphs for binary labelling and thus each move space

has $\mathcal{O}(2^n)$ allowed moves from any labelling configuration f. Other move spaces in this same category include the relabel and jump move spaces.

For the case where the smoothness prior is a semi-metric function, α-β-swap moves can be used, and if the smoothness prior is strictly metric, either α-β-swap or the more efficient α-expansion algorithm (with better optimality properties) can be used [44].

Swap move space:

1. start with arbitrary labelling f
2. set success := 0
3. for each pair of labels $\{\alpha, \beta\} \subset \mathcal{L}$
   (a) find $\hat{f} = \arg\min E(f')$ among all f' within one α-β-swap of f
   (b) if $E(\hat{f}) < E(f)$, set $f := \hat{f}$ and success := 1
4. if success = 1 then goto 2
5. return f

Expansion move space:

1. start with arbitrary labelling f
2. set success := 0
3. for each label $\alpha \in \mathcal{L}$
   (a) find $\hat{f} = \arg\min E(f')$ among all f' within one α-expansion of f
   (b) if $E(\hat{f}) < E(f)$, set $f := \hat{f}$ and success := 1
4. if success = 1 then goto 2
5. return f

Listing 4.1: Algorithms to find the local minimum in the swap and expansion move spaces.

4.8.1 Swap

For any set of two labels $\{\alpha, \beta\} \subset \mathcal{L}$ a swap move can be performed. Within this move space, a single move is equivalent to changing pixels in the labelling configuration f which are currently labelled as either α or β, to have as their new labels either α or β. Any new configuration could thus have swapped some α labels for β, and vice versa. The result of a single swap move is illustrated in figure 4.7(c). When considering the overall algorithm that uses the swap move space (see listing 4.1), it is not very efficient in terms of the number of labels as it requires a minimization for every possible pair of labels (per iteration).

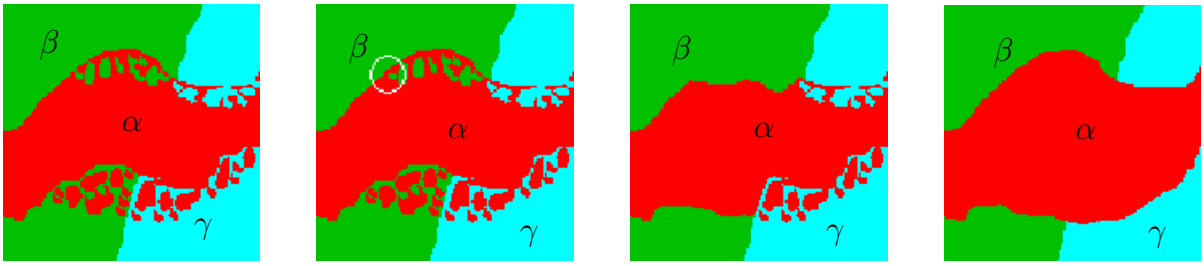
4.8.2 Expansion

An expansion move can be performed with any label $\alpha \in \mathcal{L}$. The problem is then still defined in a binary labelling sense by having a single move that changes any pixels in the current labelling configuration f to either become α or $\bar{\alpha}$. The label $\bar{\alpha}$ then merely represents that a pixel's label remains unchanged. The result of a single expansion move is illustrated in figure 4.7(d). When considering the overall algorithm that uses the expansion move space (see listing 4.1), it is relatively efficient in terms of the number of labels as it requires only one minimization for every label (per iteration). It can also alter the labels for any pixels in the whole image (as opposed to just a subset with certain labels), which allows the algorithm to make larger changes and have better optimality properties.

4.8.3 Optimality properties of α-expansion

In [44] it is proven that for a piecewise smooth prior the expansion move space produces a solution within a factor from the optimal. More specifically, with the local minimum found by expansion providing a configuration f^e and the optimal configuration being f^*, the solution's energy cost will be within a factor of $2c$ from the optimal, such that

$$E(f^*) \leq E(f^e) \leq 2cE(f^*), \tag{4.23}$$



(a) initial labelling **(b)** standard move **(c)** α-β-swap **(d)** α-expansion

Figure 4.7: An initial labelling configuration (a) consists of three labels. This illustration compares the possible results of different move spaces when (b) a standard single pixel change is made or (c) when an α-β-swap move or (d) an α-expansion move is made. Image source: [55].

where c depends on the smoothness prior and is

$$c = \max_{\{p,q\}\in\mathcal{N}} \left(\frac{\max_{l_1 \neq l_2 \in \mathcal{L}} V_{\{p,q\}}(l_1, l_2)}{\min_{l_3 \neq l_4 \in \mathcal{L}} V_{\{p,q\}}(l_3, l_4)} \right). \tag{4.24}$$

In the case of a piecewise constant prior,

$$\max_{l_1 \neq l_2 \in \mathcal{L}} V_{\{p,q\}}(l_1, l_2) = \min_{l_3 \neq l_4 \in \mathcal{L}} V_{\{p,q\}}(l_3, l_4) \tag{4.25}$$

and so $c = 1$, which gives

$$E(f^*) \leq E(f^e) \leq 2E(f^*). \tag{4.26}$$

This means that the resulting energy cost will be within a factor of 2 from the optimal energy cost.

From the theory in this chapter it can be seen that the class of energy functions considered is highly applicable to computer vision problems and that they can also be optimized to great effect by using graph cuts. In the next chapter considerations are given for developing the approaches and details surrounding the energy function that would be required to deal with the problem of the planar segmentation of range images.

Chapter 5

Graph cut segmentation

While multi-label graph cut optimization (GCO) has been applied to many different problems in computer vision, a study of the literature shows that it has not yet been used to attempt segmentation of range images. In this chapter methods are presented and discussed for using GCO to achieve the goal of segmenting a range image into planar regions.

5.1 Formulation as a labelling problem

To use GCO to solve the problem of segmenting a range image, the problem first needs to be formulated as a labelling problem. Thus, a choice needs to be made considering which properties will be represented by the sites and labels. While an attempt will be made to set up the GCO to directly provide a segmentation, it would also be sufficient to set it up such that it provides useful information that can lead to a full segmentation.

Either way, to formulate the problem it is seen that sites are naturally chosen as spatial concepts such as pixels or regions, or as detected features such as edges or corners. When choosing sites as regions (i.e. multiple connected pixels) they cannot represent the final segmented regions, as those are still unknown, and the neighbourhood structure between sites needs to be known before optimization. Since the lowest level of the data is in the form of pixels, it can be argued that this is the most appropriate choice for sites. It also fits with the definition of segmentation as assigning region labels to pixels (see section 1.2.4). Defining what the labels should represent is not as straightforward and a few possibilities present themselves.

5.1.1 Labels as regions

When considering the possible properties that can be represented by labels, the most obvious choice is having labels directly represent the different regions of the segmented image. Unfortunately, there are certain constraints that limit this and other possible label choices. Firstly, the labels are discrete and the number of labels needs to be known before optimization, but the number of required regions is unknown a priori. However, this is only a problem if the number of labels is chosen too low, so having many labels should alleviate the problem (although at the cost of possibly increasing the solution space unnecessarily). The other constraints are not related to the general labelling problem, but rather to the constraints on the energy function that graph cuts can optimize.

Since the data cost can only be defined for assigning possible labels to individual sites, it becomes impossible to assign a data cost that is dependent on the current labelling configuration unless the energy function changes dynamically during optimization. This is because, in graph cut

optimization, all the values for each possible term need to be precomputed so that they can become weights in the graphs that will be used for optimization. This is not a problem in most cases, but if the labels represent unique planar regions then the most reasonable data cost of assigning a label to a pixel would be the distance between the pixel's spatial location and a least squares plane fitted through that region. As the plane can only be fitted once the region is known, the energy function will need to be altered according to the current labelling configuration and this does not seem possible with the current graph cut optimization technique.

5.1.2 Labels as planes

There are other options for label representations that are less direct, but would still provide a segmentation as a result. This idea can be particularly useful since the case is being considered where each region is part of a planar surface. If labels were to represent the different possible planes, then setting up the data cost would not be problematic as it could simply be the distance between the site's spatial position and the chosen plane. However, since a discrete and finite set of labels is necessary the infinite number of possible planes cannot be represented. There would also be problems in determining the similarity between different planes. Some solutions to these problems would likely require a large amount of labels, which is not ideal.

5.1.3 Labels as per-pixel properties

A further option can be found when looking at the problem from a different perspective. The graph cut optimization does not necessarily need to provide a segmentation directly. The strength of the technique in handling noise can be used to restore some property that would be helpful in determining a segmentation. From experience with range images, the noise in the surface normals seems to be the greatest barrier to successfully separate planes (as the examples throughout chapter 3 have indicated). If GCO is to be used to restore the normals, then setting up a data cost would fit neatly into the form of the energy function, but the problem remains that there are an infinite number of possible surface normals for which it is necessary to define a discrete set of representative labels. Fortunately the problem is less pronounced with surface normals than with possible planes, as planes require at least three continuous parameters to be defined, while normals require only two.

In fact, it is found that working with surface normals is quite ideal because if the surface normals are correctly reconstructed for an image consisting of only planar surfaces, then it would be expected that all the normals belonging to a single surface are identical and connected. Thus, most of the regions would already be segmented. This would not be true for the cases where adjacent planes are parallel, but at different depths, because these would be seen as the same region when considering only surface normals. However, as is shown later, this problem is easily overcome.

We choose to explore this option and thus have sites as pixels and labels as surface normals. In the following sections it will be considered how to fit this choice into a GCO framework.

5.2 Discretizing normals

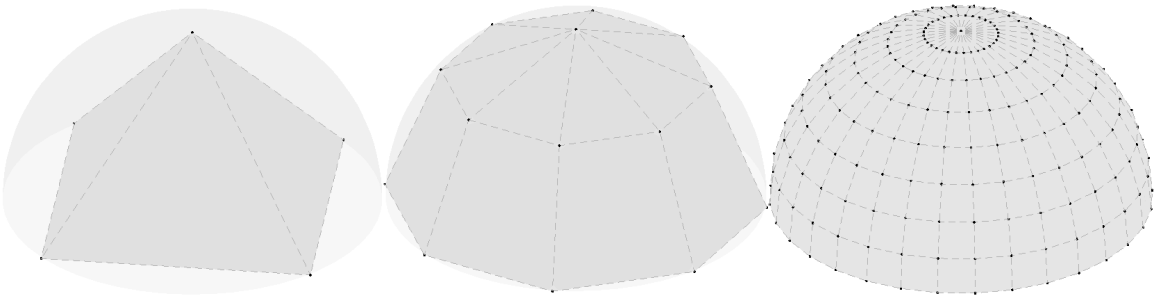
Next the choice of label representation is defined more specifically. This includes how labels should be mapped to represent surface normals and how many labels to use. It is necessary to reduce the continuous space of possible normals to a finite set of labels. The continuous parameters that represent normal vectors, θ and ϕ for azimuth and elevation (see section 3.1.2),

therefore need to discretized and mapped to a set of labels $\mathcal{L}$. Recall that all surface normals are calculated to point towards (and not away from) the image sensor and the parameters are thus limited to the intervals $0 \leq \theta < 2\pi$ and $0 < \phi \leq \frac{\pi}{2}$. The space of possible normals is therefore bounded to points on the surface of a unit hemisphere, making the problem equivalent to considering various methods for distributing points on this surface.

It is assumed that all normal directions are equally likely and equally noisy and that the most appropriate distribution is thus to uniformly distribute points across the hemisphere surface. The validity of this assumption is discussed further in section 5.9.

5.2.1 Regular angle distribution

A simple way of distributing the points is first considered. This is done by considering the two parameters θ and ϕ as axes that need to be divided regularly in a grid-like fashion. Thus, points are spaced regularly along the θ and ϕ axes by choosing an angle Δ that will be the smallest angle between points. Both axes are divided by the Δ value to determine the locations of the positions on these axes. This method results in the type of distribution shown in figure 5.1. As can be seen, normals appear closer to one another as ϕ increases, and at $\phi = \frac{\pi}{2}$ numerous points lie on top of each other. This simplistic method fails to distribute the points uniformly, and achieving this is in fact non-trivial.



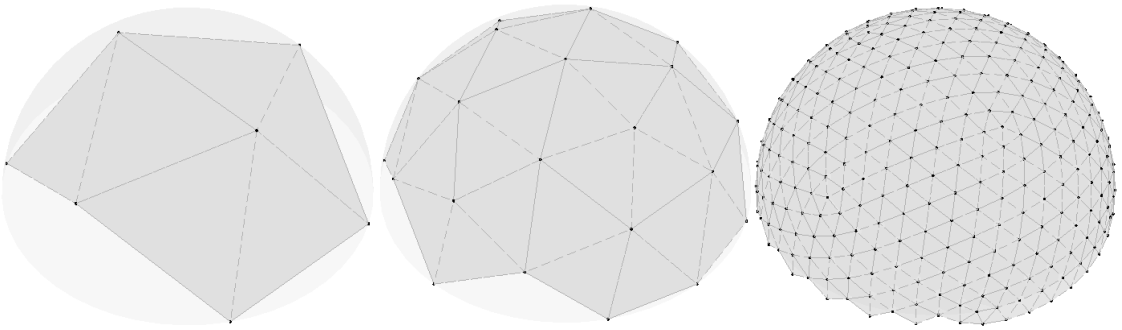
(a) $\Delta = 90^\circ$ **(b)** $\Delta = 45^\circ$ **(c)** $\Delta = 11.25^\circ$

Figure 5.1: The distribution of points on the hemisphere when the elevation and azimuth axes are divided uniformly by an angle Δ. The total number of points in each distribution is (a) 5, (b) 17 and (c) 257. The number of points excluding those at $\phi = 0$ are (a) 1, (b) 9 and (c) 225.

A variety of methods have been developed for distributing points uniformly on the surface of a sphere, many providing different results as they depend on different ways of how the problem is formally defined [56]. One definition is to consider the points as charges that repulse one another and to find a stable configuration where the repulsion between neighbouring points is balanced, generally by simulating the problem. Another definition is to maximize the minimum distance between any two points, where either the spherical or Euclidean distance (in the 3-dimensional space) is considered. The optimal arrangements for this definition are known for some specific numbers of points ($N = 2, 3, 6, 12, 24$) and for arbitrary numbers of points the arrangement can be approximated with methods such as hypercube rejection [57], spiral placement [56] or Platonic subdivision. The hypercube rejection method chooses points uniformly within a cube circumscribing the sphere and points outside of the sphere are rejected while those inside are scaled to lie on the surface of the sphere. Spiral placement places points in a spiral pattern from the top of the sphere sequentially at equal distances. Platonic subdivision takes a Platonic solid and subdivides its surfaces and rescales the points to form a tessellated sphere with a geodesic grid. In the following section this approach is considered.

5.2.2 Platonic subdivision

The Platonic solids are convex regular polyhedra, which means that they consist of faces that are congruent regular polygons and have the same number of faces meeting at every vertex. It is known that there are only five Platonic solids: the tetrahedron, hexahedron (cube), octahedron, dodecahedron and icosahedron, which have polygon faces that are either triangles, squares or pentagons. An icosahedron is initially used because it has the most points (12), while also consisting of triangular faces. Triangular faces are preferred as their points are all equally spaced from one another. Loop subdivision is then used to subdivide the triangular faces of the icosahedron by dividing each triangle into four subtriangles through the addition of new points in the middle of each edge [58]. These new points are then scaled to lie on the unit sphere and the resulting polyhedron is then divided in half for the distribution of points on the hemisphere. The number of points on the hemisphere depends on where the polyhedron is divided. All that remains is to choose a specific symmetry plane (discussed in appendix A.2), and the results are visible in figure 5.2.



(a) icosahedron **(b)** one subdivision **(c)** three subdivisions

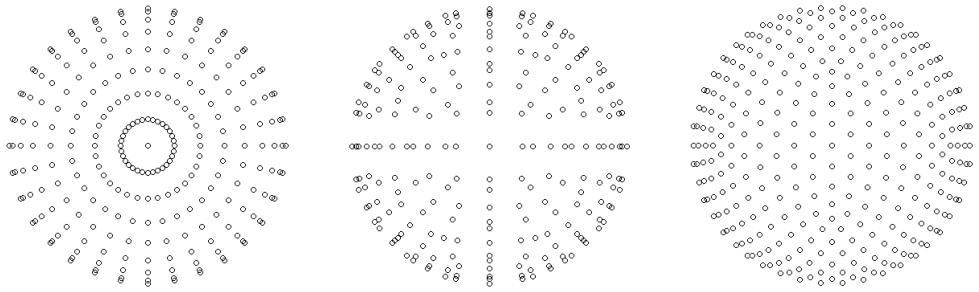
Figure 5.2: The distribution of points on the hemisphere when an icosahedron is subdivided with Loop subdivision to generate new vertices. The total number of points in each distribution is (a) 8, (b) 25 and (c) 337. The number of points excluding those at $\phi = 0$ are (a) 4, (b) 17 and (c) 305.

As can be seen, this method provides a much more even distribution (a comparative view can be found in figure 5.3). The distribution is slightly jagged along the bottom edge where the even distribution has been disrupted by selecting only points lying on one hemisphere. An interesting observation [56] is that optimal distributions with many points tend towards points that lie in hexagonal formations, but with a few hexagons distorted to fit onto the sphere. This same effect is visible in the subdivided polyhedra, where the triangulation causes most points to have a hexagonal neighbourhood, while the original points of the icosahedron maintain a pentagonal neighbourhood structure.

When considering how many points to use in the distribution it can be seen that more points allow more normals to be represented accurately, but too many points lead to an abundance of labels that would negatively affect the computational performance of the graph cut optimization. The number of points generated with either regular angles or Platonic subdivision cannot be chosen precisely (the possible values are derived in appendices A.1 and A.2), but it is found that this is not too much of a limiting factor.

5.3 Ordering labels

When choosing labels to assign to each of these discretized surface normals it is necessary to consider how the ordering of the labels will affect the smoothness function in the graph cut



(a) regular angles **(b)** hypercube rejection **(c)** Platonic subdivision

Figure 5.3: A comparison of distribution results when viewed from above. The number of points are (a) 257, (b) 298 and (c) 337. Hypercube rejection is used more often with random points and is found useful as it easily scales to higher dimensions.

optimization framework. A smoothness function that is metric would be preferred as it allows more move spaces to be usable (see section 4.8). When choosing labels as surface normals, the smoothness function will need to base the similarity between labels on the similarity between surface normals which depends only on the angular difference between them.

An exception to this is when a piecewise constant prior, where different labels are considered to be equally dissimilar, is used. This will always provide a metric function. Alternatively, a semi-metric function does not require subadditivity (discussed in section 4.5.2) and a smoothness function that is a linear scaling of angular differences allows any way of assigning labels to normals and still be semi-metric. Fortunately, as we argue in the next section, a piecewise constant prior is perfectly suited and any label ordering is therefore acceptable.

5.4 Smoothness prior

In the context of planar segmentation it is known that each region should represent a surface where all normals are identical. When the further assumption is made that there is no correlation between the surface normals of neighbouring regions (except that they are different), it is found that the piecewise constant prior is a suitable smoothness prior. Recall that the piecewise constant prior defines the smoothness function as

$$V_{\{p,q\}}(f_p, f_q) = u_{\{p,q\}} \cdot [1 - \delta(f_p, f_q)]\,, \tag{5.1}$$

where

$$[1 - \delta(f_p, f_q)] = \begin{cases} 0, & \text{if } f_p = f_q, \\ 1, & \text{otherwise.} \end{cases} \tag{5.2}$$

This function weighs any difference between neighbouring labels equally which means that large regions with identical labels have a much lower cost than multiple small regions, and relabelling separate regions does not affect the smoothness cost in any way. With this function, only the $u_{\{p,q\}}$ parameter needs to be defined. This parameter will be called the neighbourhood weight as it is defined for every pair of neighbouring pixels.

5.5 Neighbourhood weights

Graph cut optimization requires the definition of a neighbourhood structure that represents which pixels are neighbours. This is done by specifying a neighbourhood weight $u_{\{p,q\}}$ for each

possible pair of pixels p and q. A value of 0 indicates no neighbourhood connection and a value of 1 is customarily used to indicate that a pair of pixels are neighbours. The usual method of assigning neighbourhood weights is considered, as well as a few variations that can be more suited to range images and the goal of segmentation. Some of these possibilities are illustrated in figure 5.4.

5.5.1 4-connectivity

4-connectivity fits perfectly with the rectangular grid structure of pixels in an image and is therefore the most commonly used neighbourhood structure. In this case a pixel is the neighbour of another if it lies above, below or to the left or right of that pixel. These pairwise connections are weighted as 1, while all other possible connections are weighted as 0.

5.5.2 Jump edge disconnection

The problem with basic 4-connectivity is that the whole graph cut optimization is based solely on the surface normals of the original image. By ignoring all other data, different planes that are adjacent but at different depths are likely to be mistakenly merged together when they happen to be parallel to each other. Since jump edges are the easiest edge types to detect (as seen in chapter 2), this flaw can mostly be eliminated through jump edge disconnection, where jump edges are detected and completely disconnected from their neighbours. Edges are detected with the techniques discussed in chapter 2 and whose results were shown in figure 2.11(c). The data is first smoothed with a median filter and edge detection is then performed with a Sobel operator. The final edges are then chosen by using a threshold technique that includes ridge thinning and by using morphological bridging for contour closure.

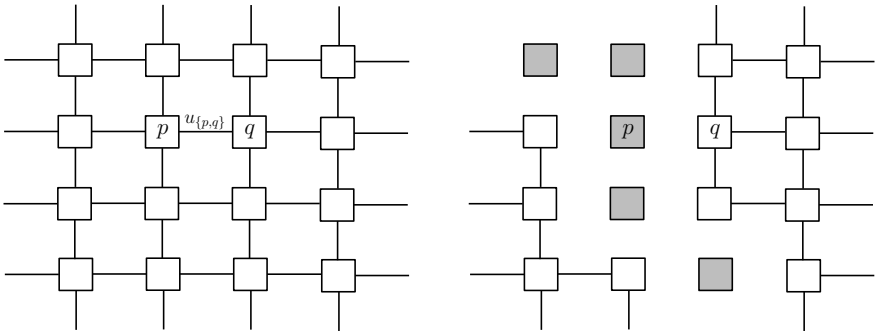
The neighbourhood weight parameter is thus used to introduce more contextual knowledge into the optimization process. It can also improve results since the graph cut energy function often has more incentive to optimize for properties of an entire region and can ignore local image properties that strongly indicate edges. This also alleviates other problems caused by the strong dependence on surface normals, e.g. that calculated surface normals tend to be noisy around jump edges.

5.5.3 Variable edge strengths

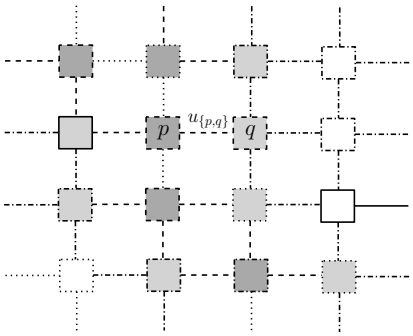
Disconnecting jump edges requires hard choices about which pixels are edges based only on the low-level data available before optimization is performed. This is a greedy approach that could force the optimization into a local optimum that carries the flaws of the initial edge detection method. A more appropriate option might be a soft approach that introduces information about the probability of a pixel being an edge, so that the hard choices about all region boundaries can still be left to the optimization process.

One complication with this approach is that edge detection methods are generally pixel-based in the sense that each pixel has a different likelihood of being an edge, such as with the edge magnitude of the Sobel operator (discussed in section 2.2.1). This means that the edge likelihoods for neighbouring pairs of pixels need to be combined in some manner. A more significant problem is that the thresholding steps of many edge detection techniques usually also alleviate other problems such as multiple edge responses being generated near an edge. Multiple edge responses would cause accuracy around edges to decrease, thereby slightly negating the goal of this approach. This method is not attempted as the concept of variable strength values can be

approached in another manner that sidesteps these problems while also taking advantage of the available contextual knowledge that is unique to range images.



(a) 4-connectivity **(b)** jump edge disconnection



(c) variable edge strengths

Figure 5.4: Some options for a neighbourhood structure include: (a) the commonly used 4-connectivity, (b) jump edge disconnection where pixels identified as jump edges are disconnected from their neighbours and (c) methods that introduce contextual information by varying the neighbourhood weights across the image. In this case (c) is a method based on variable edge strengths that produces multiple edge responses.

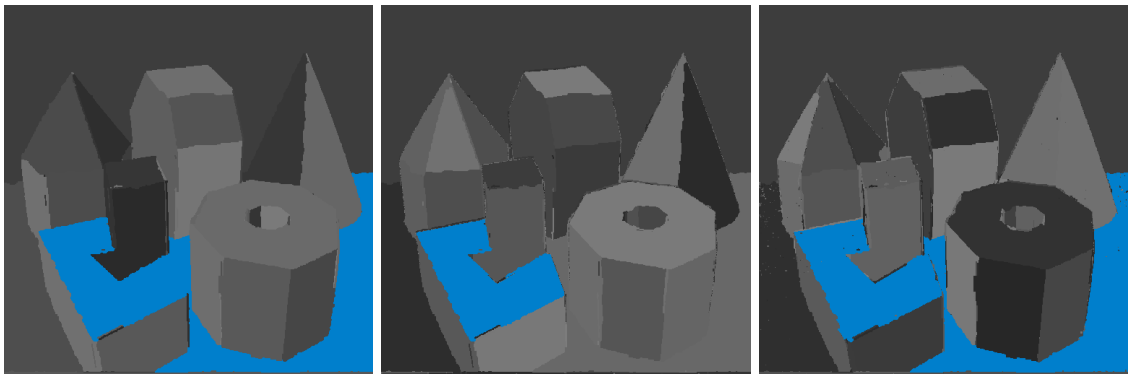
5.5.4 Distance weighting

Instead of a measure of probability where the likelihood of pixels being edges is identified, weights can be assigned according to how likely pixels are of being neighbours in the same region. Recall that pixels in a range image actually need to be transformed to 3-D points to obtain their true locations in space and in relation to one another. The Euclidean distance between adjacent points can then be used to determine weights in a manner such that points that are closer to each other have higher neighbourhood weights. This could allow the optimization process to make better decisions regarding how points should be grouped into regions.

5.5.5 Comparison

A comparison of these approaches is shown in figure 5.5 (where the complete graph cut segmentation method was used in conjunction with each of the approaches). In figure 5.5(a) the 4-connectivity approach has caused the largest floor region and another parallel region to be merged. This mistake has been fixed in figure 5.5(b) where jump edge disconnection was used. Unfortunately, the distance weighting approach shown in figure 5.5(c) did not successfully separate these regions, as there are gaps in the edge between these regions. Some extra noise is also visible with the distance weighting approach and, while the edge weights can be strengthened to insure that the correct regions are not merged, this adds extreme amounts of noise to the

results. The types of processing that are often used to find complete sets of edges are very varied and while some of them, such as edge thresholding, could be incorporated into this graph cut framework, this is definitely not the case for many of them, such as contour closure techniques. Jump edge disconnection is therefore preferred as any advanced edge detection approach can be incorporated in a simple manner.



(a) 4-connectivity **(b)** jump edge disconnection **(c)** distance weighting

Figure 5.5: The segmentation results of different weighting strategies are compared: (a) uses a simple 4-connectivity which causes the graph cut optimization to combine planes that are parallel and adjacent; (b) uses jump edge disconnection which successfully prevents this merging and also improves other jump edges; (c) attempts to use the distance between points to determine the weights, but this introduces extra noise and is not satisfactory as weak weights that have little noise do not prevent merging, while strong weights that better separate regions cause lots of noisy regions.

5.6 Data term

Recall that the energy function being optimized is

$$E(f) = \sum_{p \in \mathcal{S}} D_p(f_p) + \lambda \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p, f_q), \tag{5.3}$$

and now it only remains for us to define the data term. For each pixel in the image the data term should have an associated cost for each possible label that could be assigned to that pixel. This represents how closely a labelling configuration matches the observed data. In this case the choice is relatively straightforward. The angular difference is taken between the surface normal measured at a pixel (see section 5.7) and the surface normal represented by the label assigned to that pixel. This angular difference is the smallest angle between the normals, as measured in the plane containing both normals.

The angle θ between any two normals can be determined from the the vector dot product as

$$\theta = \arccos\left(\frac{\vec{v_1} \cdot \vec{v_2}}{\|\vec{v_1}\|\|\vec{v_2}\|}\right), \tag{5.4}$$

which can be further simplified when normals with unit length are used:

$$\theta = \arccos\left(\vec{v_1} \cdot \vec{v_2}\right). \tag{5.5}$$

The data term will thus be the sum of the angular differences between the labelling normals and the observed normals. Either radians or degrees can be used for the data cost as any scaling factor will be cancelled out when determining the value for the λ parameter (that adjusts the relative importance between the data term and smoothness prior).

5.7 Calculating observed normals

The data term requires a comparison of the current labelling configuration that is being evaluated against the normals observed in the actual data. As the normals are not directly observable in the range data, a neighbourhood structure and method to estimate these normals need to be decided upon (as also discussed in section 3.1.2).

5.7.1 4-connected cross-product

The simplest method is to take advantage of the 4-connectivity structure that is available in range images. The surface normals can then be calculated by using the four neighbours of a point to create two tangential vectors, whose cross-product is a vector normal to those vectors. While this method lacks any inherent robustness to noise, this also allows it to better represent the raw data and leave the noise handling to the optimization process.

5.7.2 Least squares plane fitting

A larger $n \times n$ neighbourhood structure, also based on the 4-connectivity structure, can be used as explained in section 3.1.2. This allows the use of more neighbouring points as well as a method that also has some robustness to noise. A least squares optimization approach can be used to fit a plane to the points in the $n \times n$ neighbourhood and the normal of the plane is then used as the normal for that point.

However, due to its increased complexity and the possibility of unwanted smoothing, we opt for the 4-connected cross-product method and, as mentioned above, leave the noise handling to the optimization process.

5.8 Relative importance of data and smoothness

Recall that equation (4.1) has a parameter, namely the λ value, which is a constant that adjusts the relative importance carried by the two terms of the energy function. The λ value is a constant that is used to scale the smoothness term. It essentially allows a choice to be made over whether more confidence should be given to the observed values of the data or to the prior assumption about the overall smoothness.

Recall that the chosen smoothness term is a piecewise constant prior which penalizes labelling configurations according to the number of neighbouring labels that are not the same. High values of λ will weigh the smoothness term more strongly, meaning that it contributes more to the final energy value. This would mean that the total energy function can be better minimized by focusing on minimizing the smoothness term, which would generally lead to larger regions. This effect is seen in figure 5.6 where the approach was run multiple times on a single image, while only changing the λ parameter (the specific approach is the one discussed in section 5.11 and using the regular angle distribution).

In figure 5.6(b), where $\lambda = 0$, the smoothness terms carries no weight at all and the optimal configuration is to fully trust the data term. The extreme noise in the labelling result corresponds to the noise of the normals calculated directly from the data. Fortunately, while the normals are very noisy, there is an underlying pattern that can be observed as some regions have a lighter or darker average colour than others. It is this information that will hopefully be extracted in the optimization process. In images (c) to (f) it is clear that as λ increases and the smoothness term

carries more weight, the resulting regions increase in size. However, while the average region size increases the data term is still always kept in consideration and small regions are still quite possible. When λ is increased to a very high value, as in (f), then the resulting image would usually have only one region but in this case, where the approach used jump edge disconnection, some region boundaries can never be crossed and some structure is still maintained at high values of λ.

The choice of λ depends on several factors such as the choices of data and smoothness terms, the noise level inherent in the data and even the structure of the data. For instance, if the data term uses radians instead of degrees then the data term will have a smaller value overall and for similar results the λ value should be smaller as well. The fact that the λ values used in figure 5.6 specifically seem most acceptable within the range of 30 to 100 is significantly influenced by the choice to use degrees to represent the difference between normals. Also consider that different neighbourhood structures would mean that the overall smoothness penalty can vary significantly. It is generally hard to reason about a specifically appropriate λ value and we therefore opt to determine it empirically, as explained in the next chapter.

5.9 Noise in the normals

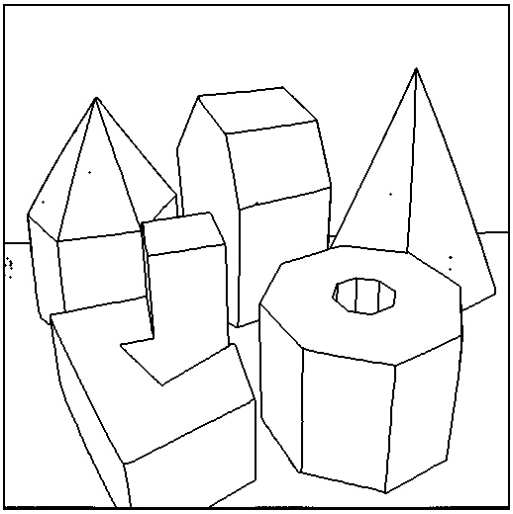
When considering all the images in figure 5.6, it is notable that some regions are cleaner than others for the same value of λ. For instance, with $\lambda = 30$ the wall in the background is already clearly identified while the floor is still very messy and unrecognizable. It can be seen that the lighter planes (where normals point more towards the viewer) are less noisy than the others and this behaviour might indicate that normals pointing in certain directions might inherently have more noise. As λ is chosen still higher, the image starts looking clearly segmented although some noisy regions remain. These regions seem to be caused by extra noise that is present along the edges of the surfaces and due to this they seem to have thin line-like shapes.

Both these noise patterns could be attributed to the capturing process where a laser range finder, which uses interferometry, was used. More noise around the edges and in regions slanted away from the sensor do seem physically plausible. It can be explained by considering that more of the laser signal is being deflected away and that a weaker signal, more prone to noise, is therefore being reflected back to the sensor.

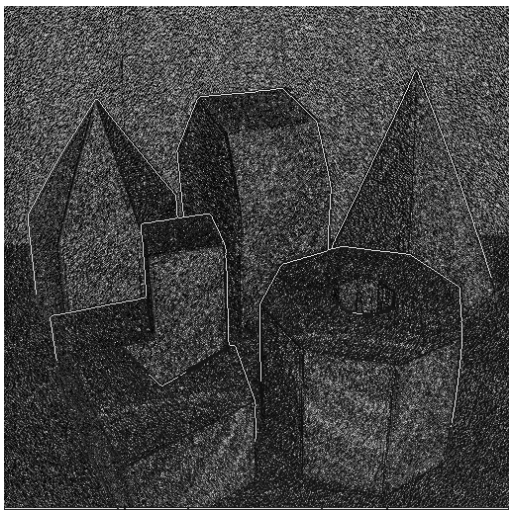
While this seems to invalidate the assumption that normals are equally noisy in all directions, it is an aspect that seems specific to the capturing process and it is preferable to hold this assumption to keep the approach more generally applicable to a variety of capturing techniques. This aspect will however be considered during the evaluation of our approach.

5.10 Converting a labelling to a segmentation

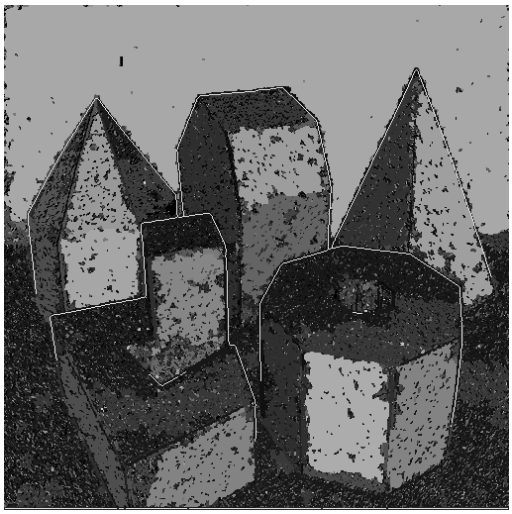
Once the optimization has been performed it will provide a labelling configuration that represents a reconstruction of the normals in the scene before they were corrupted by noise (as seen in figure 5.6), and this output can be used with the assumption that the scene consists of large regions with identical normals caused by planar structures. However, this labelling configuration is not yet a segmentation as these labels represents the normals of the regions and could be the same for separate regions. As a final step connected component labelling (as discussed in section 3.1.1) is used to relabel all the regions in the image to form a segmentation.



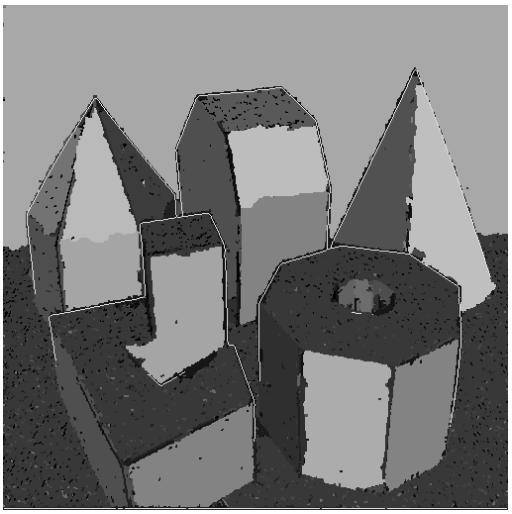
(a) ground truth



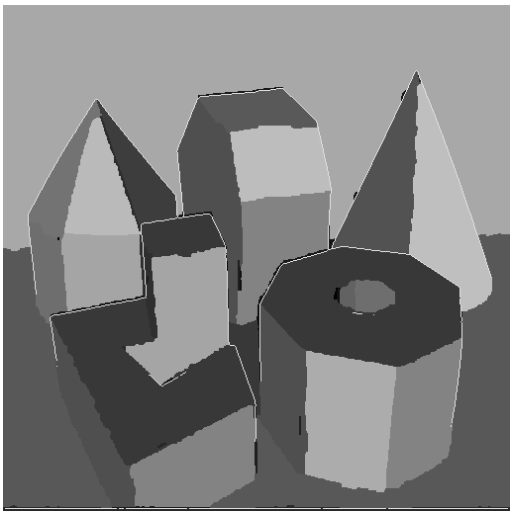
(b) $\lambda = 0$



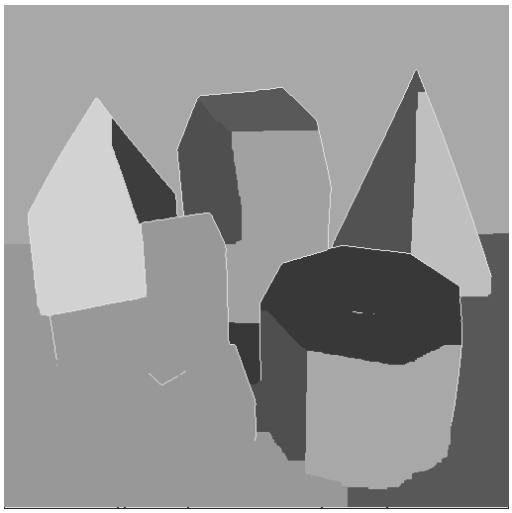
(c) $\lambda = 30$



(d) $\lambda = 40$



(e) $\lambda = 75$



(f) $\lambda = 310$

Figure 5.6: A comparison of the graph cut optimization results for different values of λ. Here the colours of the regions are in grey-scale according to the label values, with higher values corresponding to lighter regions. The order of the labels is such that normals pointing directly towards the sensor are lighter than those pointing away. These results use jump edge disconnection (seen as the white regions along some of the edges) and discretized the normals with a regular angle distribution using 225 normals. This is the same approach used in our paper [59].

5.11 Summary of algorithm

It has been discussed how graph cut optimization can be used to perform a segmentation, and some possible variations of this approach have been considered. The structure of the algorithm is outlined in figure 5.7 and consists of the following steps.

1. Taken as input is a range image and scaling algorithm that transforms depth data to a metric 3-D point cloud (as discussed in section 1.2.2).
2. Sites are defined as pixels in the range image, and labels are discretized surface normals on the unit hemisphere. Exactly what type of discretization, and how many labels to use, is determined empirically in a training phase as outlined in section 6.3.
3. An observed normal vector is calculated for every pixel in the range image, using the metric 3-D data. It was argued in section 5.7 that the use of a simple 4-connected cross-product method can be well suited for this graph cuts approach.
4. The data term is specified as the sum of angular differences between observed normals and the current label configuration (as explained in section 5.6).
5. Putative jump edges are identified through the use of Sobel operators and morphological bridging (as was done for the example in figure 2.11).
6. A neighbourhood structure over the pixels is specified. The standard 4-connectivity structure is employed, as explained in section 5.5.1, with jump edge disconnection (see the motivation given in section 5.5.5).
7. The smoothness term is chosen to be piecewise constant, as explained in section 5.4, taking into account the neighbourhood structure.
8. The λ value, used to set the relative importance between data and smoothness, is obtained empirically through training in section 6.5.2.
9. The fully specified energy function given in equation (5.3) is optimized with α-expansion (as explained in section 4.8.2).
10. The optimal labelling found through α-expansion is converted to a segmentation through simple connected component labelling as mentioned in section 5.10.
11. Finally, a simple post-processing step may be added to improve results. Motivation and detail are given in sections 6.1.4 and 6.5.

Some of these steps have used a qualitative reasoning backed by one or two sample images to choose which variations of the approach to follow, but many of the other options are not as clear cut. The approach decided upon uses jump edge disconnection for the neighbourhood weights and the cross-product method to calculate the observed normals. For the other options, such as how to discretize the normals and which λ values to use, a quantitative approach will be used as discussed in the next chapter.

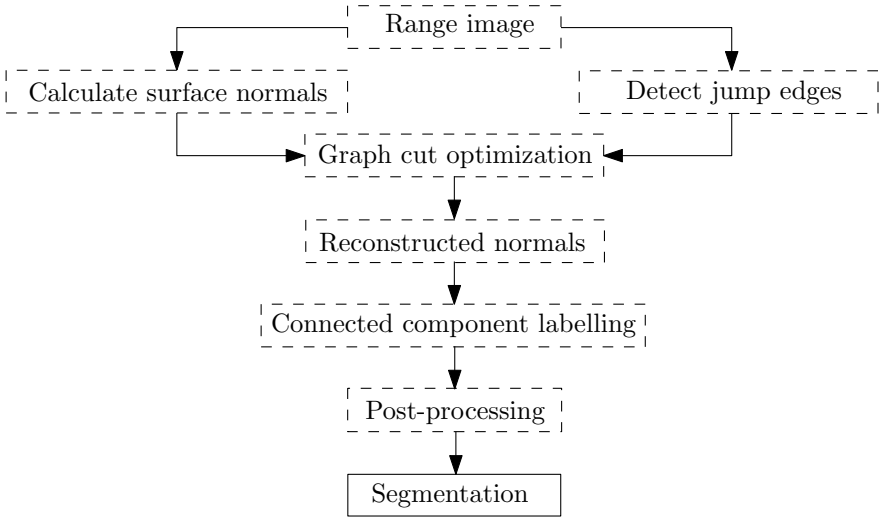


Figure 5.7: An outline of our approach.

Chapter 6

Experimental results

Until now possible options and variations of the proposed approach have been considered by considering if they seem to improve the results based on single images. This qualitative approach is useful when the improvements are obvious, but not when they are subtle and possibly ambiguous in different situations. A quantitative evaluation is necessary for these types of choices, as well as for comparing against other approaches found in the literature. In this chapter the results of applying the method on a training dataset are considered with a quantitative evaluation to choose which variant approach and parameters to use. The proposed approach is then compared, with all parameters and choices fixed, against various other approaches found in the literature by evaluating the results on a common test dataset. Discussion will first focus on the evaluation framework that will be used.

6.1 Quantitative evaluation

Algorithms in the field of computer vision are often qualitatively evaluated by comparing the appearance of their results against one another. While this is useful for a quick cursory comparison, it can be very subjective and practical solutions often require a more quantitative evaluation that can strictly define how well, and to what extent, a method fails or succeeds under certain conditions.

Fortunately, in the case of range image segmentation, a framework for comparing algorithm results against ground truth data has been created by Hoover et al. [2]. This framework includes range image datasets along with ground truth segmentations and a methodology that quantitatively compares machine segmentations with the ground truth by measuring five formally defined performance metrics. While their framework was designed to handle general segmentations, their initial work started by considering only planar segmentation [2], with some follow-up work also adding datasets containing curved surfaces [3, 7].

6.1.1 Datasets

The planar data consists of two datasets, one taken with a laser range finder (the Perceptron set) and the other with a structure light sensor (the ABW set). Both datasets consist of 40 images with resolution 512×512 and a bit depth of 8 and 12 bits for the ABW and Perceptron sensors, respectively. Each of the images consists of a scene containing a number of polyhedral shapes that have been placed in various positions and orientations. Since the polyhedral objects lie on a flat floor with a flat wall behind them, the entire scene consists of only planar surfaces. In the

case of the structured light sensor, the image data also contains shadow areas where no data was captured, as happens when object parts obstruct the light used during the capturing process.

Both datasets are divided into a training set containing 10 images, and a test set that contains 30 images. Any training or parameter tuning should be done on the training dataset, and a full comparative evaluation can then performed on the test dataset. Parameter tuning is done differently for different algorithms and is often done manually, although some work has been done to include automated parameter tuning into the framework and it has been found to perform similarly or even better than manual tuning [8, 3].

6.1.2 Performance metrics

The five performance metrics used for a full comparison emerge when classifying each region as either correct, over-segmented, under-segmented, noise or missing. These metrics are measured by considering the amount of overlap in region areas between a machine segmentation (MS) and the ground truth (GT). A threshold value T is required to define how strictly areas need to match to be classified into one of these categories. This threshold value is a percentage of the pixels that overlap between regions and falls within the range $0.5 < T \leq 1.0$.

With a pair of regions, R_m from the MS and R_g from the GT, and the number of pixels in each region being P_m and P_g, a *correct* classification occurs when the number of overlapping pixels, P_{mg}, satisfies these two conditions:

$$\begin{aligned} P_{mg} &\geq T \times P_m, \\ P_{mg} &\geq T \times P_g. \end{aligned} \tag{6.1}$$

Thus, a region in the MS is classified as correct if the area of overlap with a region in the GT consists of the majority of both regions.

Over-segmentation occurs when a single region in the GT has been segmented into multiple regions in the MS. Hence, with multiple regions $R_{m_1}, \ldots, R_{m_x}$, with $x \geq 2$, in the MS and a single region R_g in the GT, an over-segmentation is detected when the following criteria are met:

$$\begin{aligned} P_{mg} &\geq T \times P_{m_i}, \ \forall i \in [1, \ldots, x], \\ \sum_{i=1}^{x} P_{m_i g} &\geq T \times P_g. \end{aligned} \tag{6.2}$$

These two criteria specify that the majority of each of the many MS regions should overlap with region R_g and that, when combined, these regions should overlap with the majority of R_g. *Under-segmentation* is then clearly the opposite and occurs when the MS has mistakenly used a single region to describe many regions in the GT. It is defined similarly, with only the many regions and the single region reversed.

Regions that are not classified as correct, over- or under-segmented are then classified as either *noise* or *missing*. Noise regions are found only in the MS and correspond to spurious region detections. Missing regions occur only in the GT and are the regions that have been missed by the MS. In the cases of correct, over- and under-segmented regions, it is possible for a region to match the criteria for more than one of these classifications. Since each classification has two overlapping criteria, these cases are then decided in favour of the classification whose criteria have the highest average overlap.

While these metrics insure that no region will go unclassified in either the MS or GT, they cannot guarantee unique classifications for $T < 1.0$. For this reason, comparisons are always shown for multiple values of T. While the number of correct regions identified is often the focal point of

comparison, the authors of [2] have specifically defined multiple metrics as they feel strongly that a single value to evaluate accuracy or precision is often not enough, especially since certain error metrics and failure modes are more relevant to certain applications. For instance, when segmenting range images for navigation it could be more important to keep the number of missed regions as low as possible since a vehicle could easily collide with an unseen region (while a noisy region would just cause it to be over-cautious or take an alternative route). Conversely, in an application such as a bin picking task where a type of object should be selected from a container containing many objects and objects types, missed regions could lead to parts being overlooked, while noisy regions could lead to incorrect parts being selected.

6.1.3 Criticism

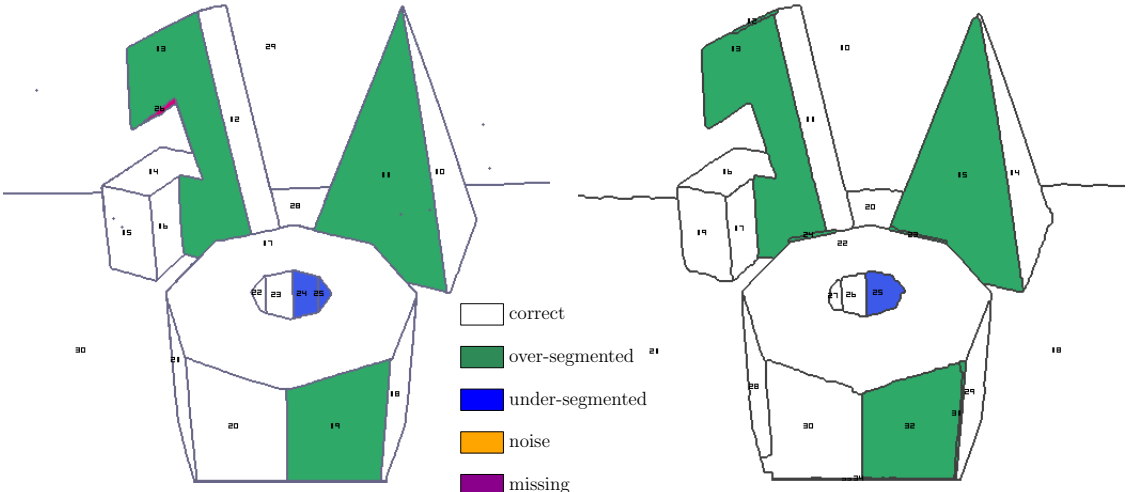
The ideal for introducing such a quantitative evaluation framework is that it becomes easier to determine the state-of-the-art and that the design of algorithms and the practical implementations can be separated from each other. This would mean that some researchers can focus on designing segmentation algorithms and, as long as they present their algorithms along with the performance results in this evaluation, it would then be easy for others to decide which algorithm they should consider when solving their practical problem. The evaluation is of course more applicable when datasets are used that are closely related to the intended application. A more surreptitious problem is that this leads to algorithms that are tailored according to the strengths and flaws of the evaluation methodology.

Unfortunately the evaluation methodology is not perfect, especially because results that qualitatively appear to be good can still provide poor results with the above definitions of the performance metrics. Some types of errors can cause much larger classification differences than expected even if in many applicable problems those types of errors might be insignificant or easily handled. For instance, consider figure 6.1 that shows the results of a comparison between the GT data and an MS. The inherent noise along the edges of the data have caused the MS to introduce small regions along some edges. Ideally these regions would be identified as noise and would not influence the number of correct regions, especially since they are so small and the comparison threshold is so low ($T = 0.51$). While this is usually the case, it is unfortunately dependent on the location of these region and will not happen when these regions lie completely within one of the GT regions. What then happens instead is that whichever region they lie in is always classified as being over-segmented into at least two segments. This leads to two over-segmented regions, instead of one correct region and one noisy region. The cause of this seems to be related to how classifications are handled when a region meets the criteria of multiple classifications. In this case the combinations of the two regions in the MS match the region in the GT better than any of those regions separately, which leads to a decision in favour of an over-segmentation instead of a correct segmentation, even though enough criteria are met for both.

On the other hand, it could be said that some flaws are not given enough consideration. For instance, the region at the bottom left of the octagonal cylinder in figure 6.1 has a highly disrupted edge. This error will only be found for higher values of the threshold T because only the number of correct pixels within a region is taken into account, and never the shape of region boundaries.

These anomalies can be attributed to the fact that the performance metrics consider only the percentage of overlapping area of regions, while ignoring region boundaries and the relative sizes of regions. Mostly, it is a symptom of the fact that the performance metrics are slightly abstract and are not always directly related to the requirements of the specific application for which segmentation is being performed (excluding perhaps the case where the MS is perfectly

correct). This does not subtract from the use of having such a quantitative framework, as it is only a problem when this goes unnoticed and the number of correctly identified regions is blindly trusted as the only measure of an algorithm's success.



(a) ground truth **(b)** machine segmentation

Figure 6.1: An example of the results from the evaluation framework, for $T = 0.51$. The (a) ground truth and (b) machine segmentation for the same image are shown and each region is colour coded according to how it was categorized. This example also shows how small noise regions disrupt the classification of other regions (the three green regions).

6.1.4 Tailoring our approach

While it is preferable to keep our own segmentation approach as general as possible and not tailor it to this evaluation framework, we find that a few simple post-processing steps can significantly improve our scores and better avoid some of the flaws that would unintentionally give poor scores with this evaluation framework. This is done by first identifying the regions that are most likely to be noise regions, and to then either merge these regions with their neighbours, or simply mark them as noise (a special region type) in which case the evaluation framework removes them from consideration.

The noise regions are caused by noise in the data and are generally found near or along edges where noise is greatest. With the proposed graph cut segmentation algorithm noise regions can also be found elsewhere when the λ parameter is low enough for the noisy data to carry greater weight than the smoothness prior. It is hard to detect these noise regions by their size because, while they are usually quite thin, they can be very long which means their overall area can be larger than other valid areas. However, it is found that these regions can be reliably detected by looking for thin line-like shapes. This property is quantified by calculating the ratio between the total number of pixels contained in a region (the area) and the number of pixels on the perimeter of the region (the circumference). This ratio is always greater or equal to 1 and a lower ratio implies either a very small region or a region that is thin and line-like. Through some testing it is decided to identify regions as noise when this ratio is lower than 2, i.e. when

$$\frac{area}{circumference} < 2. \tag{6.3}$$

This approach might not be applicable in general as its reliability might be biased by the current datasets that contain large regions. A major theoretical problem occurs with planes that are

heavily angled away from the sensor such that they appear only as thin lines, although these are already hard to detect for range sensors as well as other segmentation approaches.

Once these line-like regions have been identified, they can either be marked as a special type of noise region, in which case the evaluation framework completely ignores them for the purposes of determining the performance metrics, or they can be merged with surrounding regions. When merging these regions, a morphological operation will be used that iteratively erodes identified regions by changing the labels of eroded pixels to those of surrounding regions. Pixels are assigned to a surrounding region when the majority of their neighbouring pixels belong to that region. This erosion is performed until the detected noise regions have disappeared.

6.2 Implementation

The proposed range image segmentation algorithm is implemented in Matlab and uses the multi-label optimization (gco-v3.0) library developed by Olga Veksler and Andrew Delong [60]. The base approach will be defined as a graph cut optimization that calculates normals from the data with the cross-product and uses 4-connectivity and jump edge disconnection for its neighbourhood structure. These choices are made based on comparing the small sampling of qualitative results and also the theoretical discussions that have been presented thus far. We now continue to analyse the proposed approach using the quantitative evaluation framework to determine which normal discretization approach, post-processing step and λ values should be used. These approaches are all compared on a training dataset and those that perform best will be included in the final algorithm that will be run on a test dataset for comparison against other approaches in the literature.

6.3 Comparison of normal discretization approaches

Different approaches to normal discretization, as discussed in section 5.2, are compared by running the segmentation with each approach separately on the training data of the Perceptron dataset. Platonic subdivision with 17, 73, 305 and 1249 normals is compared against the regular angle distribution with 25, 225 and 1225 normals. The training dataset consists of 10 images and the results shown in figure 6.2 are the average number of correct regions identified for different threshold (T) values. These results are without any post-processing, and with whichever λ value gave the best results (mostly around 150). From figure 6.2 it is clear that the discretization approach and even the number of normals have a small impact on the final accuracy. There seems to be a slight tendency for the regular angle distribution to be better at low values of T and worse at high values, but the change is too small and could be attributed to noise. In any case, these results might not show any general tendency and could be specific to the dataset used. For the proposed approach, it is therefore decided to use Platonic subdivision as it seems theoretically more appropriate and is not biased towards any type of normal angle distribution. It also seems that any amount of normals (among these options) would be sufficient, so it is arbitrarily decided to use 305 normals, although another aspect to consider is that this choice does influence execution time.

6.4 Execution time

This study did not focus on directly improving the execution time of the proposed algorithm and, while it is likely that many parts of the Matlab implementation have much room for improvement in this regard, the time spent during execution was still dominated by the actual multi-label

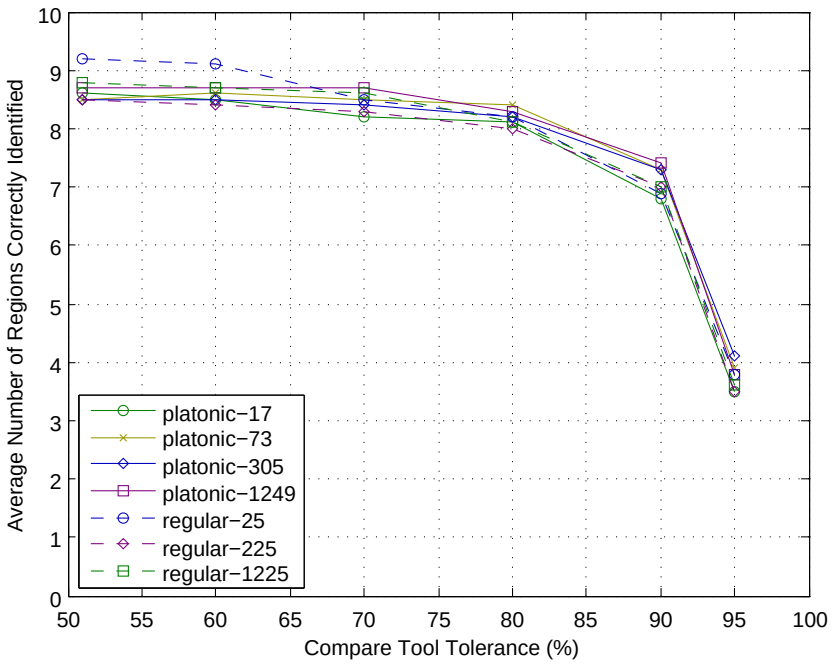


Figure 6.2: A comparison of the Perceptron training results for different normal discretization approaches (without any post-processing).

graph cut optimization which was implemented in C++ in the gco-v3.0 library. This measured execution time, as is clear from the graph cut theory, is highly influenced by the number of labels used for optimization as well as other factors such as image size and scene complexity. Results are shown in table 6.1. There is also significant variation in the execution times due to the structure and complexity of the images, and the optima that need to be found. It is also interesting to note that the average execution time appears to grow more or less linearly with the number of labels.

Platonic subdivision				
Number of labels	Mean	Variance	Minimum	Maximum
17	12	6	7	44
73	42	7	33	63
305	129	30	100	204
1249	476	112	384	809

Regular angles				
Number of labels	Mean	Variance	Minimum	Maximum
25	14	3	10	24
225	114	33	76	179
1225	657	198	414	1309

Table 6.1: These tables show the relation between the number of labels and execution times when the proposed algorithm is run on the Perceptron training dataset. All times are given in seconds.

6.5 Determining parameters and post-processing

It is now determined which post-processing and parameters to use. The only parameter is λ, which represents the trade-off between whether the result should more closely follow the observed data, in which case it should have a lower value, or whether it should be more in line with the prior assumptions on the smoothness of the data, in which case it will have a higher value.

6.5.1 Post-processing

A significant choice that still needs to be decided is which post-processing approach (discussed in section 6.1.4) to use. It needs to be decided whether to leave detected noisy line-shaped regions as they are, or whether they should be merged with their neighbouring regions, or if they should be identified as noise (a reserved region type specifically ignored by the evaluation framework*). To determine this choice the proposed algorithm was run on the training data with each of the different post-processing approaches. The results in figures 6.3(a) through (e) show how the choice of λ affects the number of regions identified as correct, over-segmented, under-segmented, missing and noise, when using the evaluation framework. The evaluation framework's threshold was chosen as $T = 80\%$ and the value of λ was varied from 20 to 150 in increments of 5. For each of these λ values a segmentation was calculated for all images in the datasets, and post-processing was performed on these results either by doing nothing, or by merely identifying noisy regions as a noise classification, or by removing the identified noisy region by merging them into their surroundings. In the case where no post-processing was performed the value of λ starts at 55 instead of 20 as the total number of unique regions in the segmentation results for lower λ values was too high for the evaluation framework to handle. The evaluation framework is unable to evaluate images that contain more than 256 unique regions, primarily due to constraints of the image storage format used for comparisons. While the correct regions detected by the no post-processing approach seems to increase with λ, it actually peaks at exactly 150 and then declines steadily like the other approaches.

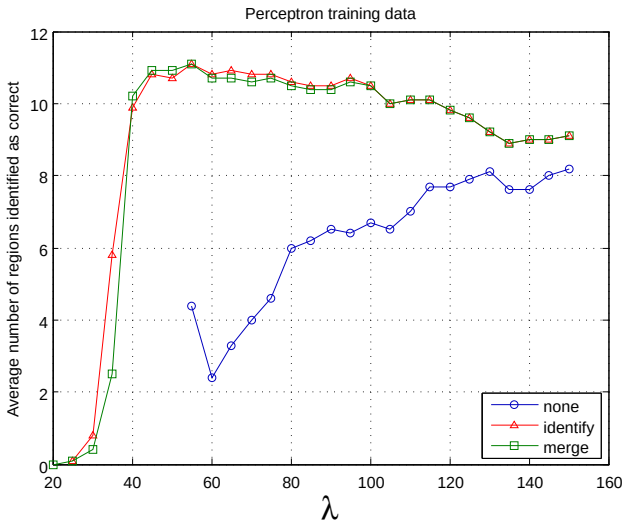
From the number of correct regions it is clear that the post-processing steps introduce a significant improvement in the overall quantitative results of the proposed algorithm, both improving the overall score as well as the stability in terms of λ. It can clearly be seen that the largest improvements caused by the post-processing are less noisy regions, which then prevent the over-segmentation effect discussed earlier and lead to more regions being identified as correct. The classifications of under-segmented and missing regions are quite unaffected by the post-processing. At higher λ values, where small noisy regions become less prominent, the post-processing steps also have much less influence. It is interesting to note that much of the stability at higher values of λ is due to the jump edge disconnection of the neighbourhood structure, as the broken edge connections prevent many regions from merging together regardless of whether it might improve the minimum energy. This prevents a lot of the under-segmentation that would occur at higher values of λ, although only for the successfully located jump edges.

It seems that the particular post-processing approach used is not that significant, as long as something is done to remove clearly incorrect regions from being considered by the evaluation framework.

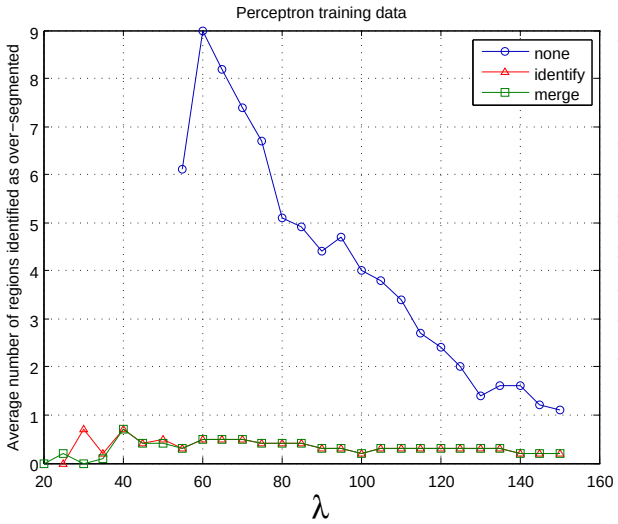
6.5.2 The relative importance parameter

As expected, higher λ values decrease over-segmentation and increase under-segmentation as the higher weighting towards the smoothness prior leads to larger regions. The number of missing, noise and even correct regions generally stays steady as soon as λ is large enough.

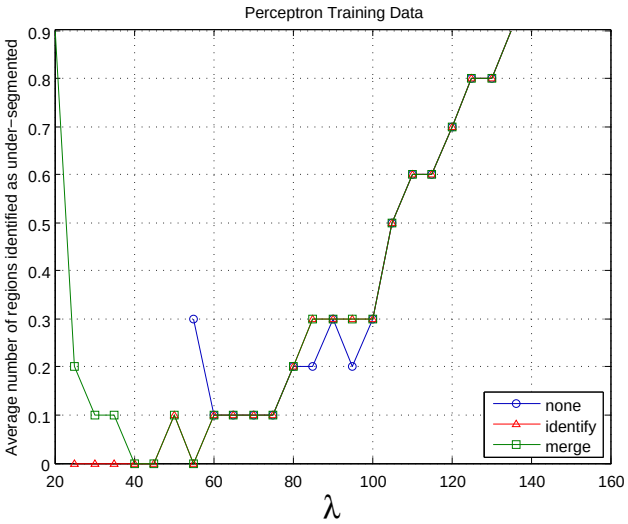
*We refer here to the regions in our results that are clearly noise and which can then be assigned to a reserved region in the segmentation results given to the evaluation framework. The discussion should not to be confused with the noise classification that the evaluation framework could assign to a region as part of its results.



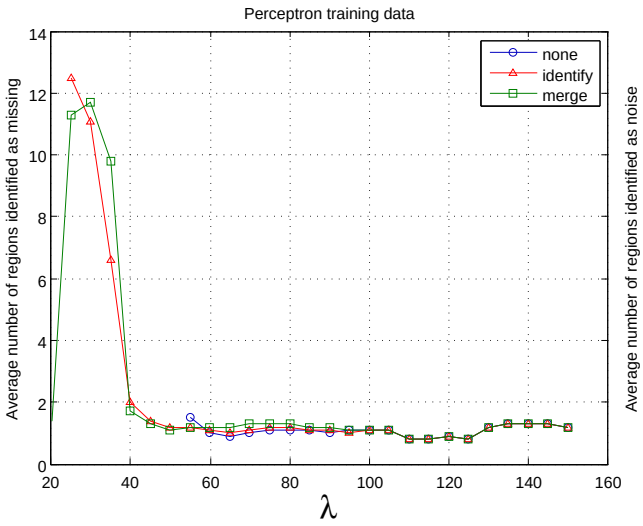
(a) correct regions



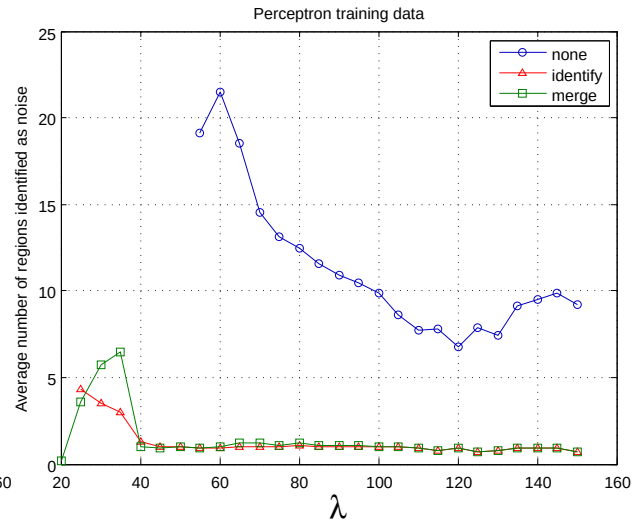
(b) over-segmented regions



(c) under-segmented regions



(d) missing regions



(e) noise regions

Figure 6.3: Comparison of how different λ values and post-processing methods affect the region classifications in the training data. The comparison tool's threshold value was set to $T = 80\%$ for all these comparisons.

When considering an ideal λ value to be used for the Perceptron dataset, it is clear that post-processing causes a large shift towards lower values of λ. The post-processing approaches seem to peak as soon as λ is high enough such that the noisy regions do not overwhelm the results. This seems reasonable as following the data closely should lead to good accuracy, especially if the noise in the data is indirectly handled by post-processing. Without post-processing the ideal λ value is always quite high as some over-segmentation becomes preferable to the extreme noise and under-segmentation of lower λ values, especially since small incorrect regions can have a disproportionate influence on the evaluation framework's results.

To fully compare the post-processing approaches, an ideal λ value is thus chosen for each approach to see how they perform in a more complete evaluation where the whole range of different tolerance threshold values are considered. The results for this are shown in figure 6.4 where the λ values were chosen as $\lambda = 150$ for the approach without post-processing, $\lambda = 55$ for the merging approach and $\lambda = 55$ for the identification approach.

The merging and identification approaches behave quite similarly, although the identification approach has a tendency to be slight better at lower threshold values, while the merging approach is better at higher threshold values. From these results it is chosen to further use the proposed algorithm with the identification post-processing step and a λ value of 55.

6.6 Comparison with other methods

With the value of λ and choice of post-processing approach fixed, the proposed algorithm was run on the full test dataset and compared against other results in the literature that used a similar evaluation methodology. The results are shown in figure 6.5 and the specific values are listed in appendix B. The segmentations achieved by the proposed approach for every test image, as well as how the comparison tool classified each region, are shown in appendix C. Perfect accuracy is obtained when all regions are classified as correct, which with the Perceptron testing dataset amounts to an average number of correct regions of 14.8. The first approaches that were evaluated using this evaluation methodology are those from the original paper by Hoover et al. [2] where universities were invited to test their approaches. These four approaches were named WSU, UB, UE and USF, according to the universities from which they originated. The USF and UE approaches use iterative region growing, while the WSU approach uses a clustering approach and the UB method uses a scan-line approach where line segments are found and grouped together. Further development based on the UB approach lead to the scan-line edge detection approach discussed in chapter 2 of this thesis, which will be called EG [11]. Some further approaches which are compared, predominately using robust methods, are RCA [61] and UFPR/OSU [40]. The UFPR/OSU approach also incorporates an interesting idea where some parameters are chosen during processing by using genetic optimization.

Overall, results from our approach compare quite favourably, although both the EG and UFPR/OSU approaches show better results. Both of these approaches have been developed and improved over a number of years, with the EG approach being based on UB, and the UFPR/OSU approach based on the UFPR [62] approach.

6.7 Other datasets

There are a few implementation details that were required to allow the proposed algorithm to be general enough to handle various different types of range images so that the proposed algorithm could be applied to other datasets. A major difference between the Perceptron dataset

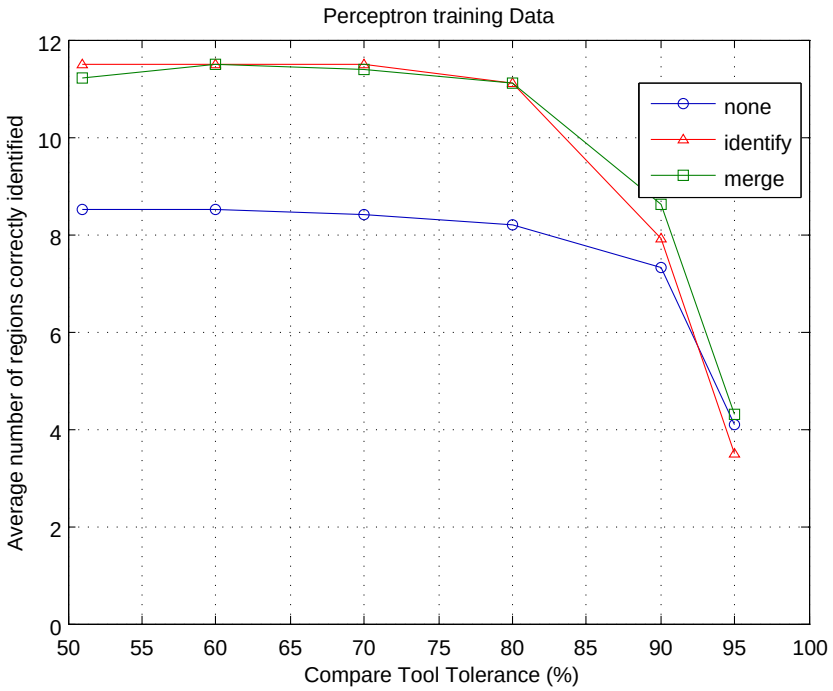


Figure 6.4: Comparison of the number of correct regions detected in the training data with different post-processing approaches and at different threshold values. Ideal λ values for each approach were chosen as: $\lambda = 150$ for no post-processing, $\lambda = 55$ for the merging approach and $\lambda = 55$ for the identification approach.

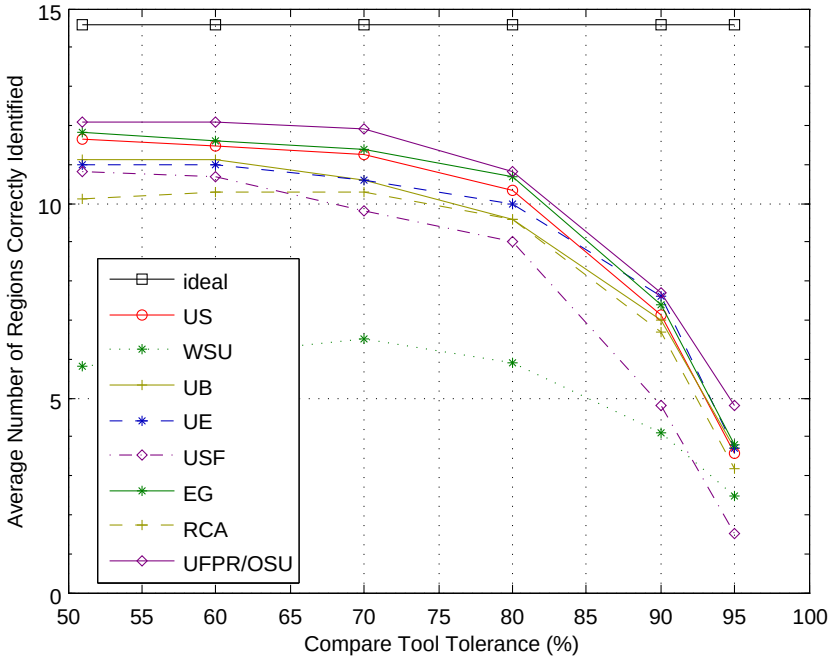


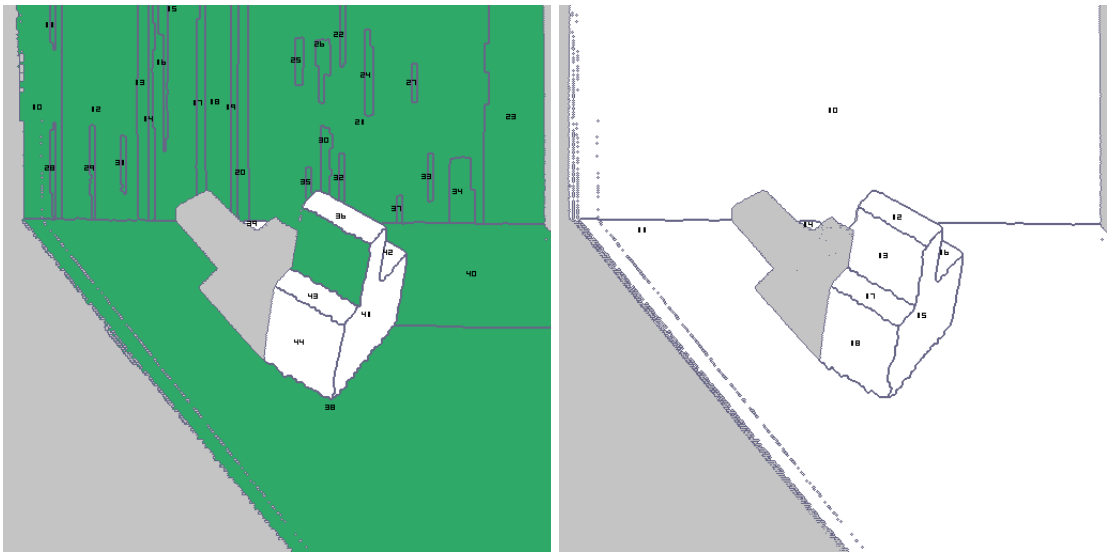
Figure 6.5: A comparison of the proposed approach (US) against other results found in the literature. The identification post-processing step was used with $\lambda = 55$. See text for more details about the other approaches.

(which was captured with a laser range finder) and some other datasets (such as those captured with structured light) is that the latter can contain shadow areas where no data was captured. To handle this, it is necessary to alter how the neighbourhood structure is set up and how edges are detected. The edge detection approach was changed in a simple manner so that it specifically ignores the abrupt changes that occur between values next to these shadow areas.

The neighbourhood structures were set up so that all shadow pixels are disconnected from their neighbours and, since no normals can be calculated at those locations, they should also not contribute any data penalties. Since the results from the graph cut optimization could thus assign any random labels to those pixels, it is therefore necessary to add an additional post-processing step that relabels points known to lie in shadow areas.

6.7.1 ABW dataset

When specifically considering applying the proposed approach to the ABW dataset by Hoover et al. (which was captured with a structured light approach), there is another problem that presents itself. While the ABW dataset is much less noisy than the Perceptron dataset, its range values are only 8-bit as opposed to the 12-bit values of the Perceptron dataset. This means that there is a much lower depth resolution in the data and it introduces much more quantization noise. This is specifically a problem as planes with very shallow angles (with regard to the x-y plane) have a significant staircase effect which means that the sloped planes seem to consist of many large flat sections. This means that, for any normal calculation approach that only consider local areas within these flat sections, the normals calculated would always be fronto-parallel. As can be seen in figure 6.6(a), this can cause the proposed approach to introduce additional unnecessary regions as it relies heavily on the normal data that is initially given to it.



(a) **(b)**

Figure 6.6: The 8-bit data of the ABW dataset introduces extreme quantization noise into the normals of planes with shallow angles, as can be seen in (a). White indicates correctly segmented regions, grey signifies shadow areas (where no depth data is available) and green indicates over-segmentation. The result in (b) shows one dramatic example of the improvement after Gaussian noise was added to the depth data to counter-act the quantization noise.

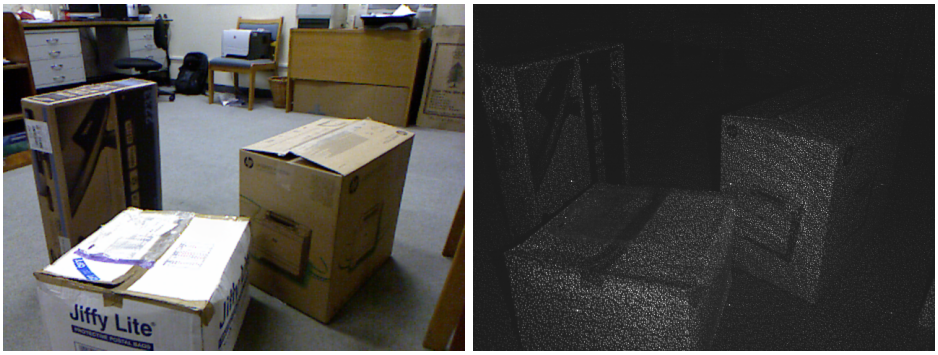
While this could be handled in a manner by using much higher λ values so that the image as a whole is much smoother, the results are still not very satisfactory. A few different smoothing techniques were attempted, and even while they did improve the depth data without affecting the edges they still often introduced too much smoothing into the normals and the edges and did not deliver satisfactory results. However, it was found that the results could be significantly improved by simply introducing a small amount of additional Gaussian noise ($\sigma = 1$) into the depth values of the points (i.e. along the z-axis), as is shown with the results in figure 6.6(b). These type of adjustments should hopefully only be necessary to correct datasets where the assumptions of the graph cut optimization have been broken. In this case, the assumption that the normals at all

points contain noise that is statistically independent was broken as the quantization noise found here is specifically related to the larger structures found in the image.

In general, other approaches perform significantly better on the ABW dataset than the Perceptron dataset, due to much lower levels of noise in the depth data. However, this is not the case for the proposed approach and in comparison to the results of other approaches the proposed approach still performs below average on the ABW dataset. This is likely a result of the artificial noise that was introduced and might signify that the proposed approach is over-reliant on the initially calculated normal data.

6.7.2 Kinect data

The Microsoft Kinect is a consumer electronics device which, among other sensors, includes a real-time structured-light range imaging sensor which was developed by PrimeSense. The sensor provides 11-bit 640×480 range images at 30 frames per second with the use of an invisible infra-red (IR) speckle pattern that is projected onto a scene and then detected with an IR camera. The distortion of the known pattern reveals the depth structure of the scene. The device's low cost, portability and accuracy in indoor environments have made it a favourite for uses on robots, with human-computer interfaces, and in various hobbyist projects. An example of images captured with the Kinect's infra-red and colour cameras is shown in figure 6.7 and some results of the proposed algorithm on Kinect data are shown in figure 6.8.



(a) **(b)**

Figure 6.7: The Kinect sensor has both (a) colour and (b) infra-red cameras which allow it to see a scene as well as the infra-red speckle pattern that is projected upon it.

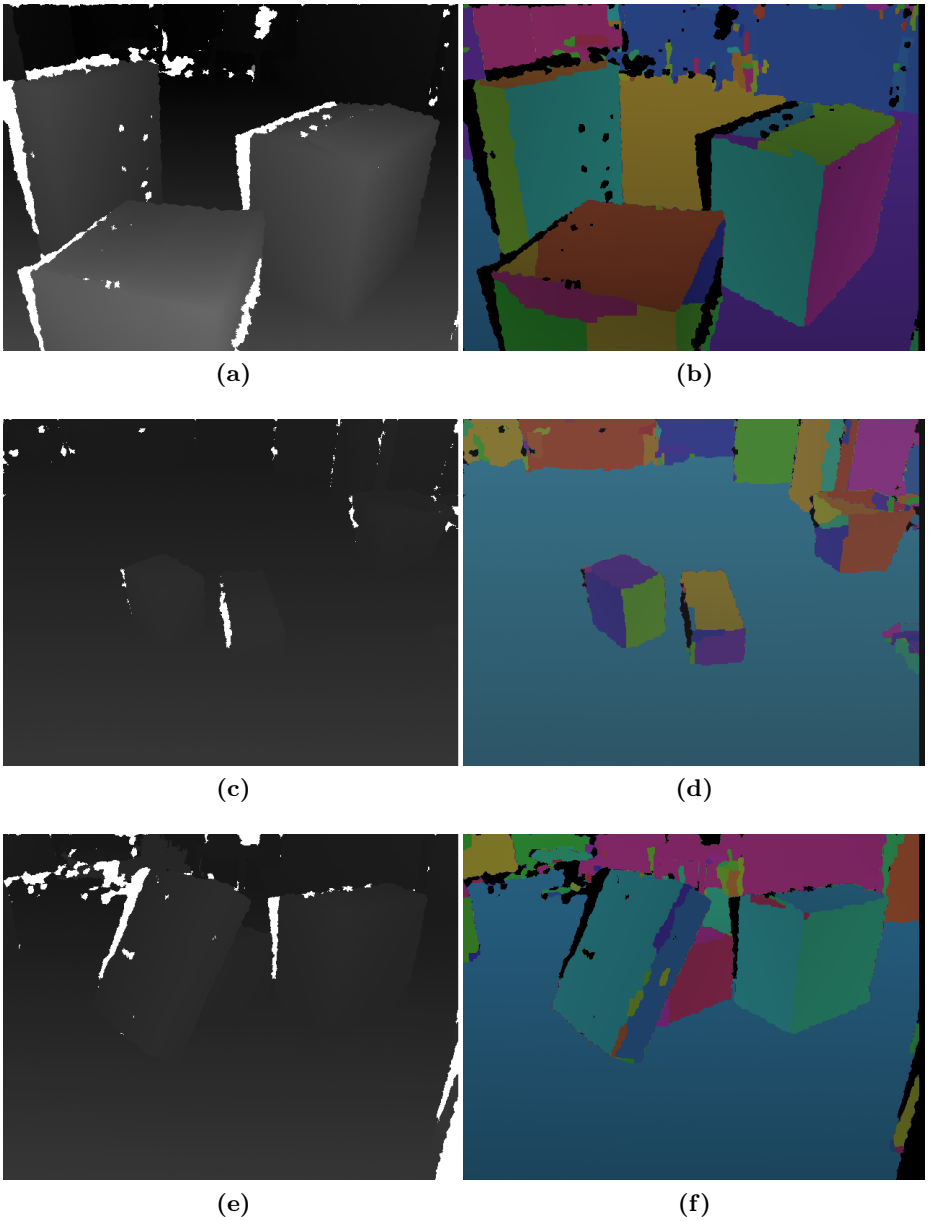


Figure 6.8: Segmentation results of the proposed approach when used on some images taken with the Kinect sensor. No post-processing steps were used and the value of λ was set to 100.

Chapter 7

Conclusions and future work

In this chapter the conclusions that can be made from the results of the proposed approach are further discussed, as well as how it compares to the other approaches in the literature. Other ways are then discussed in which graph cut optimization might be applicable to the problem of range image segmentation, and the current state of the field of range image segmentation is considered as well as how it fits into the domain of scene reconstruction.

7.1 Our results

The graph cut segmentation algorithm that has been presented has desirable properties such as having very few adjustable parameters and a relative robustness as long as the assumptions about noise hold. As we have seen in figure 6.5 the proposed approach compares quite favourably to current state-of-the-art methods, although it requires a post-processing step to handle clearly noisy regions that occur at low values of λ, as they are severely penalized by the specific evaluation framework which measures accuracy in number of regions. While some adjustment was necessary to handle the quantization noise in the ABW dataset, the promising results on both the ABW and Kinect data show that the algorithm is generally applicable and easily adjustable to different types of range images, regardless of capturing method. The λ parameter is still not fully ideal as it requires experimentation (or training) to determine, and cannot easily be pre-determined from theoretical principles. It most likely needs to be adjusted for images with different levels of noise and scenes that have different scales of structure.

7.2 Graph cut optimization

It has been found that the same graph cut optimization approach which has found so much success in the processing of colour images also has some clear uses when working with range images. The similarity between range images and standard intensity images allows many approaches to be applied to both, although it is easy to make the mistake of forgetting the differences between these image types and it might be wise to consider range images as merely a convenient format for storing point cloud data which adheres to some specific constraints. It is especially important when considering different types of contextually specific knowledge that can be introduced into the optimization framework.

7.2.1 Execution time

While graph cut optimization is very fast considering the huge solution space that has to be traversed, it would still be too slow for any practical applications that require near real-time speeds. Improving this would require speed improvements in the multi-label optimization library that was used. The library currently uses only a single-threaded approach and the most direct potential for improvement in speed would be the development of a graph cut optimization algorithm that can make use of parallel processing.

7.2.2 Applying graph cut optimization to general segmentation

It is also important to consider whether the graph cut optimization function that has been considered could be applied as a more general range image segmentation approach that is not limited to planes. This would require the capability to use various types of surface models, which does not seem possible due to the data term (recall equation (4.4)) which is limited to assigning data costs between a site and its observed value (as opposed to a group of sites and a potential model being fitted to those sites). The possibility of using a smoothness prior such as the piecewise smooth prior might allow non-planar regions to be incorporated into our normal reconstruction approach, although additional region merging would be necessary to form a proper segmentation.

There are also many other ways in which graph cut optimization can be used to handle the problem of range image segmentation. For instance, while this study focused on one type of energy function that can be optimized with graph cuts, there are actually more possible variations in terms of the energy function [43]. Variations such as adding an additional term constraining the total number of different labels in an image [63] or approaches such as ratio cuts and normalized cuts [64], that also consider region shape properties, all have the potential to improve segmentation results.

7.2.3 Segmentation as a binary labelling problem

The multi-label graph cut optimization approach which was considered does not seem to be perfectly suited for the general segmentation problem, as any segmentation could require an arbitrary number of labels that is unknown ahead of time. However, this is only the case when regions are formed by assigning labels to pixels. An alternative, which we have not seen in the literature, is to rather assign labels to the neighbourhood structure between pixels. This could probably be referred to as an edge-based approach.

This approach would entail constructing a graph from the neighbourhood structure of an image, and then assigning labels to the edges of this graph, instead of to the pixels (the nodes of the graph). If the labels assigned to these edges correspond to whether the neighbouring pixels belong to the same region, then only two labels are required to represent any segmentation. This is advantageous as graph cut optimization can better handle binary labelling problems (in terms of speed and optimality guarantees). Unfortunately, while this type of labelling can represent all possible segmentations, it can also represent invalid configurations that do not correspond to a segmentation. See figure 7.1 for some examples of valid and invalid configurations with this formulation.

When considering how to represent only valid segmentations, it can be seen that they always consist of edges that are connected in a manner that can form a curve that separates pixels into disjoint sets. This curve can be drawn in a systematic manner by considering the dual of the neighbourhood graph structure, as shown in figure 7.2. Further considering these possible curves,

there is actually an underlying neighbourhood structure that is present between the edges of the original neighbourhood structure. This is shown in figure 7.3 where it becomes clear that the edges in a 4-connected neighbourhood structures have themselves got a neighbourhood structure that consists of cliques of size 4. This is somewhat of a problem for graph cut optimization, as higher order cliques present increased complexity.

Therefore, while this idea holds some promise in representing any segmentations as a binary labelling problem, it would also include many difficulties if an energy function needs to be set up such that only valid segmentations can be delivered. However, if these obstacles could be overcome, then extremely fast segmentations would be achievable with graph cut optimization and it is therefore believed that it should be considered as an avenue for future research.

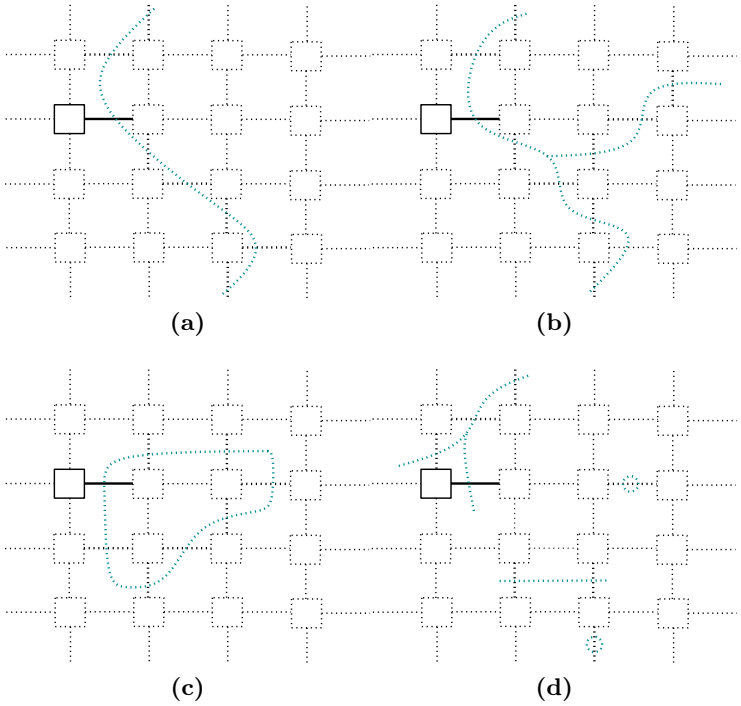


Figure 7.1: Labelling edges in a binary manner to represent the region connectivity between pixels can correspond to any possible valid segmentation, such as those seen in (a), (b) and (c). Unfortunately, arbitrarily labelling edges in this manner can also represent invalid configurations that do not correspond to segmentations, such as seen in (d).

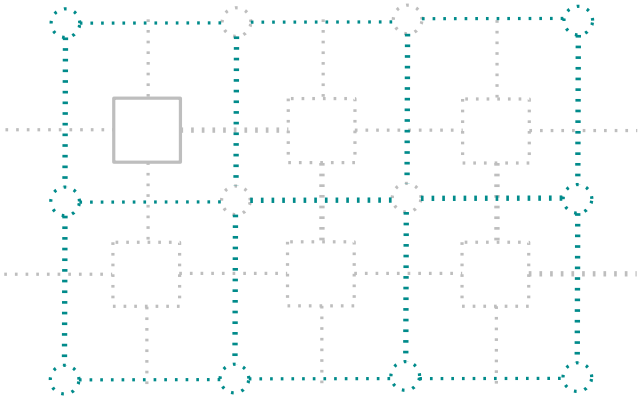
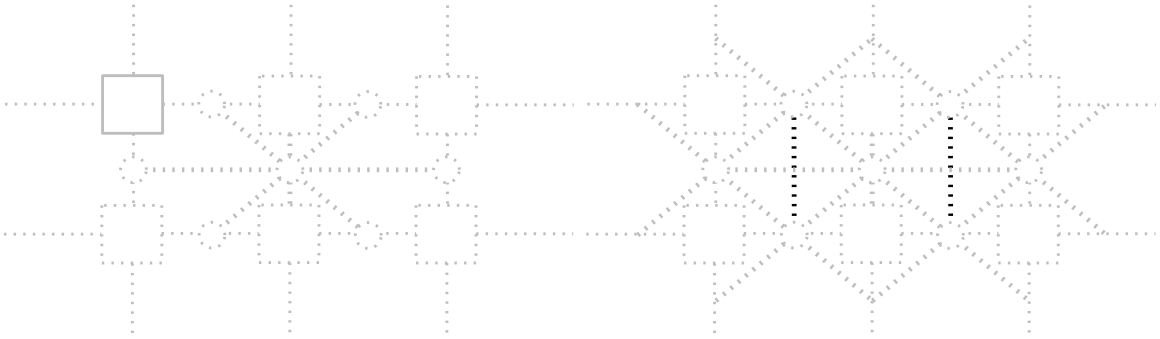


Figure 7.2: The curves separating disjoint sets of pixels (segmented regions), can be drawn by drawing connected edges on the dual of the neighbourhood structure. Here a 4-connected neighbourhood is shown, as well as its dual, which is again a 4-connected neighbourhood.



(a) (b)

Figure 7.3: The edges of a 4-connected neighbourhood structure of an image have themselves got a neighbourhood structure when only valid segmentations (that separate all pixels into disjoint sets) are considered. The neighbours of a single edge are shown in (a) and the neighbourhood structure as a whole can be seen in (b). It is clear that this neighbourhood structure contains cliques of size 4 which would complicate a graph cut optimization.

7.3 Categorizing range image segmentation approaches

This study has covered various approaches that have been used to segment range images into planar surfaces. While these approaches have often been categorized by separating them into region-based and edge-based approaches, another categorization that has become apparent is that recent approaches tend towards model-based approaches instead of the earlier data-based approaches. Data-based approaches look for certain types of structures and patterns in the data that are then combined in a bottom-up sense to find a solution, while model-based approaches try to fit various types of models to large parts of the data and are thus a top-down approach. Data-based approaches tend to be much faster to process, but can easily be disrupted by noise in the data which is often compensated for by adding many intricate processing steps that also add complexity. Model-based approaches have become favoured for their robustness to noise and intuitive simplicity in their representation of the data, but often need to traverse a large solution space and they require that the data can be explained by a small number of possible models, especially since each additional model added to such a framework increases the size of the solution space and hence the computation time.

7.4 Evaluation of range image segmentation

As discussed in the previous chapter, it seems necessary to consider that some improvements could be made to the current predominant evaluation framework [2]. It is hard to formulate a framework that evaluates segmentation results in a completely theoretical manner without considering which practical applications and requirements the segmentation algorithm needs to fulfil, although the authors of the framework specifically consider this when they mention that a single value representing the correctness of a segmentation should be avoided. However, another consideration that makes evaluating segmentations much harder is scale-space theory which finds that the structure of a scene is inseparable from the current scale that is being considered [65]. For instance, when segmenting a scene containing a large building the scale of interest would define whether it is sufficient to segment the building as separate from the background, or segment the walls and roof of the building as separate surfaces, or even whether the tiles on the roof should be segmented from one another. Each of these delivers completely different results and in the scale considering walls and rooftops the smaller structures such as roof tiles and windows would be

noise that worsen the final results. It might therefore be necessary for the evaluation framework to also include a scale of interest, although this would first require datasets that are complex enough to introduce these considerations.

It barely needs to be said that more datasets that also contain ground truth data would always be beneficial to the field of range image segmentation. While scene reconstruction of colour images can often measure ground truth data by simultaneously capturing range data, capturing ground truth data for the range data itself is not possible and ground truths need to be created either manually by people, or by fully knowing the actual structure of the scene of which the range data is being captured. For real-world scenes this is hard to achieve, but advances in the realism of graphical simulations could allow endless data to be created along with ground truths, which could allow the number of possible approaches in range image segmentation to flourish.

7.5 Scene reconstruction

The purpose of segmentation is often as a stepping stone to a more complete reconstruction of a scene, but a question to consider in the quest to improve segmentation results is whether the perfect segmentation of a scene is possible without more complete knowledge of the scene, such as a full reconstruction. This is an idea that was considered in [66] where a system was proposed that would integrate both segmentation and object representation in order to produce an improved result. For instance, in the planar segmentations considered by us, the edges between intersecting planes will always be straight lines. A segmentation approach could be improved to also take into account that certain edge shapes are more likely than others, although this would be at the risk of making the segmentation approach less general and harder to adapt for non-planar surfaces. On the other hand, a scene reconstruction has to consider which physical configurations would produce the image being processed, and concepts such as straight line edges would be naturally incorporated as no other edges could be possible in the reconstruction of intersecting planes.

When considering the segmentation approaches compared in figure 6.5, their results were compared against the ground truth data which was manually created by people in the following manner. The boundary of surfaces was considered on a per-pixel basis by creating a trace in a magnified window and alongside registered intensity and reflectance images, CAD models of the objects in the scenes and the numerical range data. As a person's visual system fully reconstructs the likely nature of the scene whenever the whole image is viewed, they have much more information available to make intelligent decisions about where edges might lie in the image. It remains an open question whether all this information is necessary to successfully segment a scene and conversely whether a segmentation is a necessary step to a reconstruction. It could likely be that the answer lies somewhere in-between, with the perfect reconstruction and segmentation only possible when both results are combined and used to improve each other in a co-operative manner.

Appendix A

Number of discrete normals

A.1 Regular angle division

We consider the number of points distributed onto a hemisphere when elevation and azimuth axes are used and points are placed at locations that are at regularly spaced angles along both these axes (as described in section 5.2.1). In a hemisphere the elevation axis is bounded so that $0 \leq \phi \leq \frac{\pi}{2}$ and the azimuth axis is bounded such that $0 \leq \theta < 2\pi$. We distribute points on the hemisphere by dividing both axes by the same angle Δ. If the number of divisions of the elevation axis is n, then the azimuth axis will be divided into $4n$ divisions. The elevation axis will thus have $n+1$ points, but since the azimuth axis is circular (with the first and last points being the same) it will have $4n$ points. The total number of points will therefore be the product of the number of points along both axes, except that the $4n$ points that lie on top of one another at $\phi = \frac{\pi}{2}$ should be counted as a single point. The total number of points is then

$$4n \times (n+1) - 4n + 1 = 4n^2 + 1.$$

If the axes are chosen to ignore points at $\phi = 0$, then the elevation axis will have only n points and there will be a total of

$$4n \times n - 4n + 1 = (2n-1)^2$$

points. The total number of points for different values of n or Δ are shown in table A.1.

n	Δ	Total points ($\phi \geq 0$)	Total points ($\phi > 0$)
1	90°	5	1
2	45°	17	9
3	30°	37	25
4	22.5°	65	49
8	11.25°	257	225
18	5°	1297	1225

Table A.1: This table shows some examples of the amount of points in a hemisphere when the elevation and azimuth axes are divided regularly. Surface normals coinciding with points at $\phi = 0$ can be ignored as they are not found in range images.

A.2 Loop subdivision of an icosahedron

We consider how the vertices in the resulting polyhedron would change when Loop subdivision is successively performed on an icosahedron (as described in section 5.2.2). Euler's polyhedron theorem (the Euler characteristic) states that any convex polyhedron's surface has the characteristic

that

$$V - E + F = 2,$$

where V, E and F are respectively the number of vertices, edges and faces of the polyhedron. An icosahedron is a regular polyhedron that has identical equilateral triangular faces. It has the property that five triangular faces meet at each vertex and its values are $V = 12$, $E = 30$ and $F = 20$. A Loop subdivision [58] would divide each triangle into four subtriangles by placing a new vertex in the middle of each edge. Thus, after one Loop subdivision the number of faces would change such that

$$F_{\text{new}} = 4F.$$

Each old edge becomes two edges, while three new edges are added for each old face, hence

$$E_{\text{new}} = 2E + 3F,$$

and to satisfy Euler's polyhedron theorem the number of vertices should change to

$$\begin{aligned} V_{\text{new}} &= 2 + E_{\text{new}} - F_{\text{new}} \\ &= 2 + 2E + 3F - 4F \\ &= 2 + 2E - F. \end{aligned}$$

Therefore, after each Loop subdivision the properties of the polyhedron would change as seen in table A.2.

The number of vertices that lie in one hemisphere of the polyhedron depends on where the polyhedron is divided. One method of division, that will make the vertices symmetrical in each hemisphere, is to divide the volume of the polyhedron in half with a plane that passes through one of the original icosahedron's edges (and thus also its corresponding opposite edge). The initial plane of symmetry passes through two edges and thus four points, and the subsequent subdivisions each doubles the number of points on this plane. The number of vertices in one hemisphere would therefore be half of all the vertices that do not lie precisely on the dividing surface and these values are shown in table A.3.

Subdivisions	Vertices	Edges	Faces
None	12	30	20
1	42	120	80
2	162	480	320
3	642	1920	1280
4	2562	7680	5120

Table A.2: This table shows how the number of vertices, edges and faces would change when Loop subdivision is successively performed on an icosahedron.

Subdivisions	Total vertices	Symmetry-plane vertices	Vertices per hemisphere
None	12	4	4
1	42	8	17
2	162	16	73
3	642	32	305
4	2562	64	1249

Table A.3: This tables compares the total number of vertices after each Loop subdivision with the number of vertices that lie precisely on the plane of symmetry and the number of vertices that are in each hemisphere. The choice of symmetry plane is explained in the text.

Appendix B

Evaluation results from the literature

The results from different sources in the literature that were performed on the evaluation framework of Hoover et al. [2] are listed in table B.1. These results were not always easily found in the various literature sources, and it was often necessary to estimate the values from the few graphs that were provided. The UBHam approach was not compared in our graph (figure 6.5) as their testing methodology was slightly different and they only seem to show the results after tuning parameters to the test dataset as opposed to using the training dataset for that purpose. We also did not show the original UFPR approach, and only included the improved version (UFPR/OSU).

Method	Threshold					
	51%	60%	70%	80%	90%	95%
WSU[2]	5.8	6.1	6.5	5.9	4.1	2.5
USF[2]	10.8	10.7	9.8	9.0	4.8	1.5
UB[2]	11.1	11.1	10.6	9.6	7.0	3.7
UE[2]	11.0	11.0	10.6	10.0	7.6	3.7
EG[11]	11.8	11.6	11.4	10.7	7.4	3.8
RCA[61]	10.1	10.3	10.3	9.6	6.7	3.2
UBHam[38]	11.1	11.9	11.6	11.2	8.4	4.6
UFPR[62]	12.6	12.6	12.1	11.0	8.1	4.6
UFPR/OSU[40]	12.1	12.1	11.9	10.8	7.7	4.8
US-ident	11.6	11.5	11.2	10.3	7.1	3.6
US-merge	11.1	11.0	11.0	10.3	8.0	4.9

Table B.1: A summary of some results from the literature, where results on the Perceptron test set data were available, compared to the approach proposed in this thesis (US).

Appendix C

Results on Perceptron dataset

The results of applying our complete segmentation algorithm to the Perceptron dataset are shown in figures C.2 to C.4. Each pair of images shows the ground truth to the left and the machine segmentation to the right, with their separate regions colour coded according to figure C.1. These are the results as used for figure 6.5, with the threshold value $T = 0.8$, and therefore use the identification post-processing step which introduces the areas without data. While the alternative, the merging post-processing step, produces results that might be more aesthetically pleasing to a person, the identification step allows the framework to better ignore areas whose results are known to be flawed. More aesthetic results are also achievable with higher λ values but, as discussed in chapter 6, combining lower λ values with an identification step produced better results with the evaluation framework.

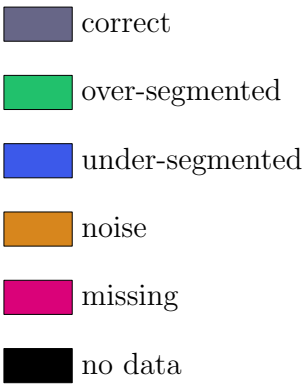


Figure C.1: Meaning of colour codes from the output of the segmentation evaluation framework [2].

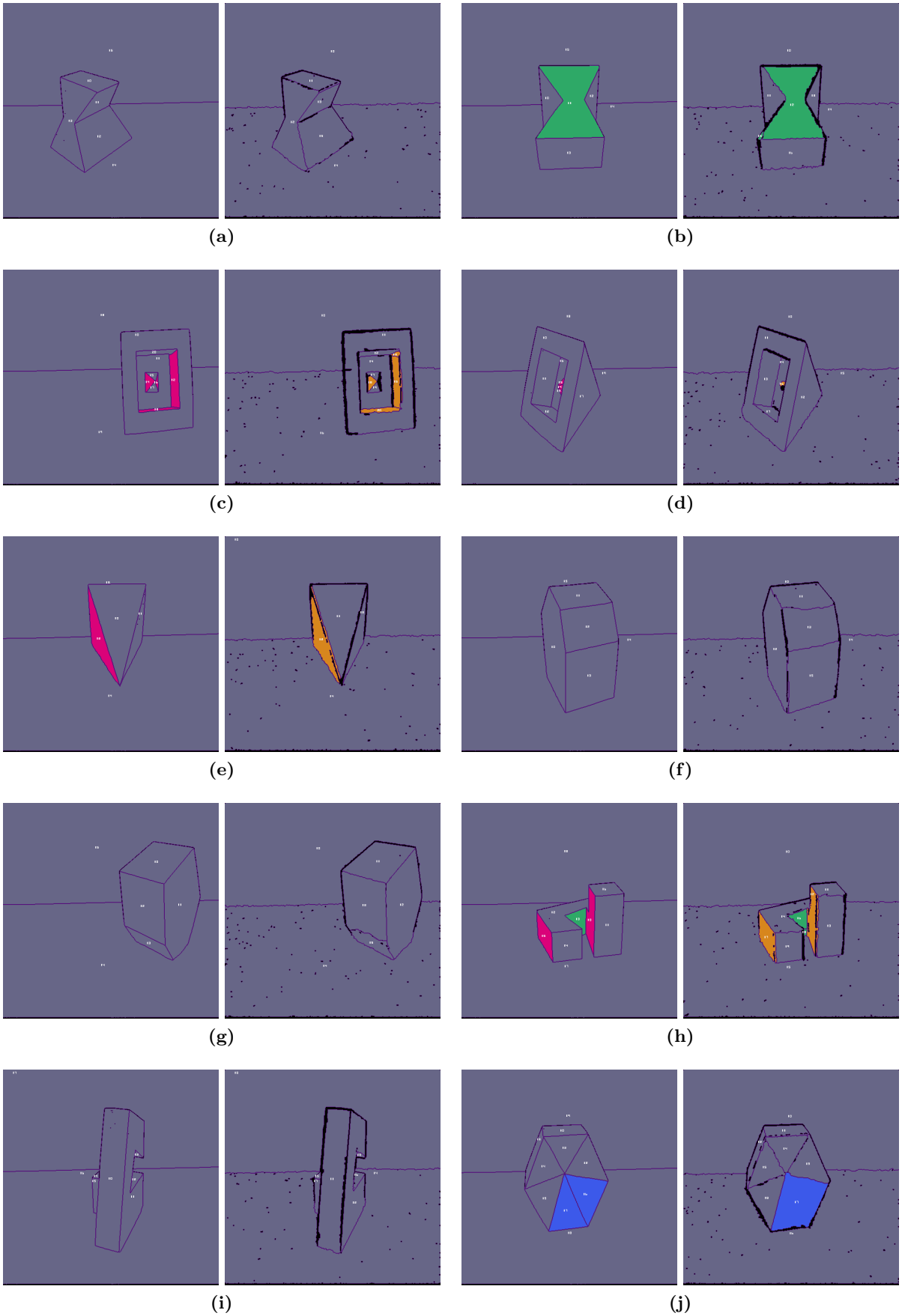


Figure C.2: Results of our segmentation algorithm when performed on the Perceptron test dataset.

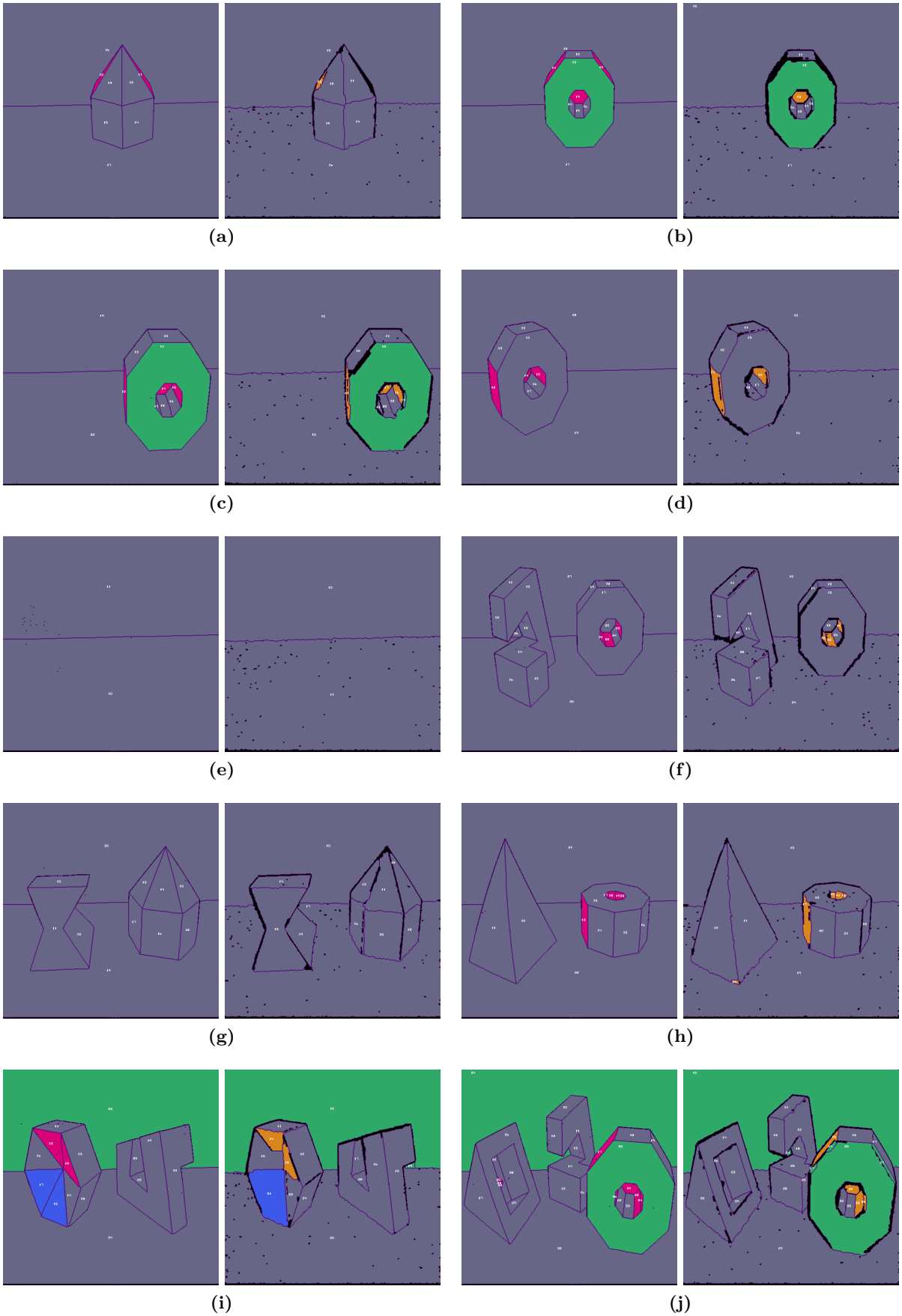


Figure C.3: Results of our segmentation algorithm when performed on the Perceptron test dataset.

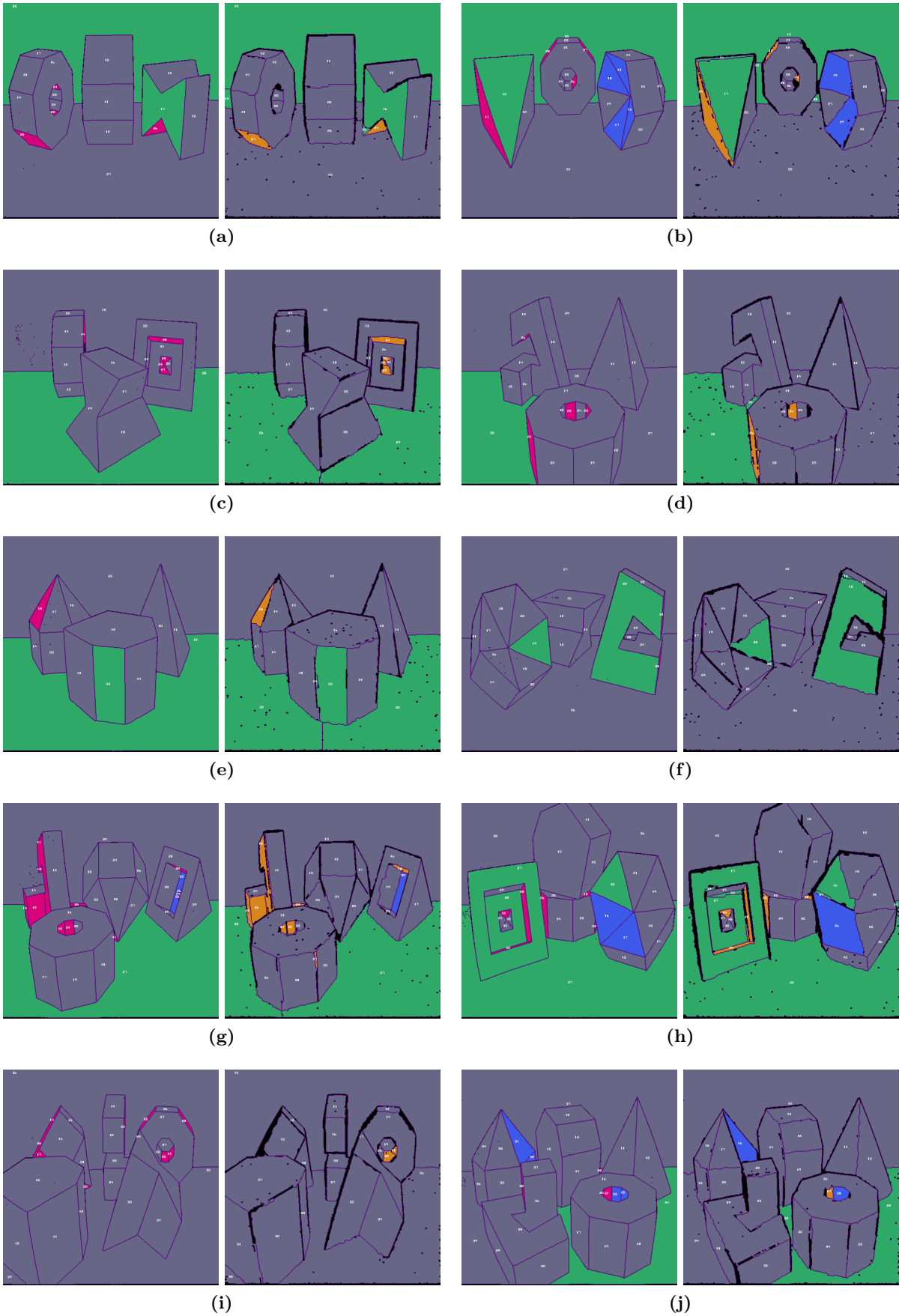


Figure C.4: Results of our segmentation algorithm when performed on the Perceptron test dataset.

Bibliography

[1] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," University of California, Tech. Rep., 2001.

[2] A. Hoover, G. Jean-Baptiste, X. Jiang, P. Flynn, H. Bunke, D. Goldof, K. Bowyer, D. Eggert, A. Fitzgibbon, and R. Fisher, "An experimental comparison of range image segmentation algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 7, pp. 673–689, 1996.

[3] J. Min, M. Powell, and K. Bowyer, "Progress in automated evaluation of curved surface range image segmentation," in *Proceedings of the International Conference on Pattern Recognition*, 2000, pp. 644–647.

[4] A.-L. Chauve, P. Labatut, and J.-P. Pons, "Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1261–1268.

[5] X. Jiang, K. Bowyer, Y. Morioka, S. Hiura, K. Sato, S. Inokuchi, M. Bock, C. Guerra, R. Loke, and J. du Buf, "Some further results of experimental comparison of range image segmentation algorithms," in *Proceedings of the International Conference on Pattern Recognition*, 2000, pp. 4877–4882.

[6] M. Powell, "Comparing curved-surface range image segmentors," Master's thesis, University of South Florida, 1997.

[7] M. Powell, K. Bowyer, X. Jiang, and H. Bunke, "Comparing curved-surface range image segmenters," in *Proceedings of the IEEE International Conference on Computer Vision*, 1997, pp. 286–291.

[8] J. Min, M. Powell, and K. Bowyer, "Automated performance evaluation of range image segmentation algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 34, no. 1, pp. 263–271, 2004.

[9] R. Hoffman and A. Jain, "Segmentation and classification of range images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 608–620, 1987.

[10] X. Jiang, "An adaptive contour closure algorithm and its experimental evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1252–1265, 2000.

[11] X. Jiang and H. Bunke, "Edge detection in range images based on scan line approximation," *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 183–199, 1999.

[12] A. Gupta, "Range image segmentation for 3-D object recognition," Master's thesis, University of Pennsylvania, 1988.

[13] R. Gonzalez and R. Woods, *Digital Image Processing*, 3rd ed. Pearson Prentice Hall, 2008.

[14] W. Pratt, *Digital Image Processing*. John Wiley & Sons, 2001.

[15] V. Koivunen and M. Pietikäinen, "Experiments with combined edge and region-based range image segmentation," in *Theory & Applications of Image Analysis: Selected Papers from the 7th Scandinavian Conference on Image Analysis*. World Scientific Publishing, 1992, pp. 162–176.

[16] M. Hebel and U. Stilla, "Pre-classification of points and segmentation of urban objects by scan line analysis of airborne LIDAR data," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, no. B3a, pp. 105–110, 2008.

[17] D. Katsoulas and A. Werber, "Edge detection in range images of piled box-like objects," in *Proceedings of the International Conference on Pattern Recognition*, 2004, pp. 80–84.

[18] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Computer Graphics and Image Processing*, vol. 1, no. 3, pp. 244–256, 1972.

[19] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *The Canadian Cartographer*, vol. 10, no. 2, pp. 112–122, 1973.

[20] X. Jiang and H. Bunke, "Fast segmentation of range images into planar regions by scan line grouping," *Machine Vision and Applications*, vol. 7, no. 2, pp. 115–122, 1994.

[21] X. Jiang, U. Meier, and H. Bunke, "Fast range image segmentation using high-level segmentation primitives," in *Proceedings of the IEEE Workshop on Applications of Computer Vision*, 1996, pp. 83–88.

[22] X. Jiang and H. Bunke, "Robust and fast edge detection and description in range images," in *Proceedings of the IAPR Workshop on Machine Vision Applications*, 1996, pp. 538–541.

[23] A. Sappa, "Unsupervised contour closure algorithm for range image edge-based segmentation," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 377–384, 2006.

[24] A. Levinshtein, C. Sminchisescu, and S. Dickinson, "Optimal contour closure by superpixel grouping," *European Conference on Computer Vision (ECCV)*, pp. 480–493, 2010.

[25] S. Wang, T. Kubota, J. Siskind, and J. Wang, "Salient closed boundary extraction with ratio contour," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 4, pp. 546–561, 2005.

[26] J. Stahl and S. Wang, "Edge grouping combining boundary and region information," *IEEE Transactions on Image Processing*, vol. 16, no. 10, pp. 2590–2606, 2007.

[27] F. Estrada and A. Jepson, "Robust boundary detection with adaptive grouping," in *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshop*, 2006, pp. 184–191.

[28] S. Mahamud, L. Williams, K. Thornber, and K. Xu, "Segmentation of multiple salient closed contours from real images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 4, pp. 433–444, 2003.

[29] A. Bab-Hadiashar and N. Gheissari, "Range image segmentation using surface selection criterion," *IEEE Transactions on Image Processing*, vol. 15, no. 7, pp. 2006–2018, 2006.

[30] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009, pp. 3206–3211.

[31] F. Schmitt and X. Chen, "Fast segmentation of range images into planar regions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1991, pp. 710–711.

[32] H. Wang and D. Suter, "MDPE: a very robust estimator for model fitting and range image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 139–166, 2004.

[33] P. Torr and A. Zisserman, "MLESAC: a new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 138–156, 2000.

[34] H. Wang and D. Suter, "A model-based range image segmentation algorithm using a novel robust estimator," in *Proceedings of the International Workshop on Statistical and Computational Theories of Vision*, 2003.

[35] G. Roth and M. Levine, "Segmentation of geometric signals using robust fitting," in *Proceedings of the International Conference on Pattern Recognition*, 1990, pp. 826–831.

[36] X. Yu, T. Bui, and A. Krzyẑak, "Robust estimation for range image segmentation and reconstruction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 530–538, 1994.

[37] K.-M. Lee, P. Meer, and R.-H. Park, "Robust adaptive segmentation of range images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 2, pp. 200–205, 1998.

[38] K. Köster and M. Spann, "MIR: an approach to robust clustering-application to range image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 5, pp. 430–444, 2000.

[39] L. Silva, O. Bellon, and P. Gotardo, "A global-to-local approach for robust range image segmentation," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 1, 2002, pp. 773–776.

[40] P. Gotardo, O. Bellon, K. Boyer, and L. Silva, "Range image segmentation into planar and quadric surfaces using an improved robust estimator and genetic algorithm," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 34, no. 6, pp. 2303–2316, 2004.

[41] S. Li, *Advances in Pattern Recognition: Markov Random Field Modeling in Image Analysis*, 3rd ed. Springer, 2009.

[42] F. Han, Z. Tu, and S.-C. Zhu, "Range image segmentation by an effective jump-diffusion method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1138–1153, 2004.

[43] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, 2004.

[44] O. Veksler, "Efficient graph-based energy minimization methods in computer vision," Ph.D. dissertation, Cornell University, 1999.

[45] D. Sedlacek and J. Zara, "Graph cut based point-cloud segmentation for polygonal reconstruction," in *Proceedings of the International Symposium on Visual Computing*, 2009, pp. 218–227.

[46] S. Tran and L. Davis, "3D surface reconstruction using graph cuts with surface constraints," in *Proceedings of the European Conference on Computer Vision*, 2006, pp. 219–231.

[47] M. Sormann, C. Zach, J. Bauer, K. Karner, and H. Bishof, "Watertight multi-view reconstruction based on volumetric graph-cuts," in *Proceedings of the Scandinavian Conference on Image Analysis*, 2007, pp. 393–402.

[48] S. Sinha, P. Mordohai, and M. Pollefeys, "Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh," in *Proceedings of the IEEE International Conference on Computer Vision*, 2007, pp. 1–8.

[49] Y. Boykov, O. Veksler, and R. Zabih, "Efficient approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1222–1239, 2001.

[50] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[51] P. Kohli, M. Kumar, and P. Torr, "$\mathcal{P}^3$ & beyond: solving energies with higher order cliques," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[52] J. Hammersley and P. Clifford, "Markov fields on finite graphs and lattices," 1971, pHS Research Grant No. GM-10525-08, National Institute of Health, Public Health Service.

[53] R. Kindermann and J. Snell, *Markov Random Fields and their Applications*. American Mathematical Society, 1980.

[54] L. Ford and D. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, no. 3, pp. 399–404, 1956.

[55] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[56] E. Saff and A. Kuijlaars, "Distributing many points on a sphere," *The Mathematical Intelligencer*, vol. 19, no. 1, pp. 5–11, 1997.

[57] P. Bourke. Distributing points on a sphere. [Online]. Available: http://paulbourke.net/geometry/circlesphere/

[58] C. Loop, "Smooth subdivision surfaces based on triangles," Master's thesis, University of Utah, 1987.

[59] S. Muller and W. Brink, "Graph cut segmentation of range images into planar regions," in *Proceedings of the Annual Symposium of the Pattern Recognition Association of South Africa*, 2011, pp. 108–113.

[60] O. Veksler, A. Delong *et al.* Computer Vision at Western (code). [Online]. Available: http://vision.csd.uwo.ca/code/

[61] H. Frigui and R. Krishnapuram, "A robust competitive clustering algorithm with applications in computer vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 450–465, 1999.

[62] P. Gotardo, O. Bellon, and L. Silva, "Range image segmentation by surface extraction using an improved robust estimator," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2003, pp. 11–33.

[63] A. Delong, A. Osokin, H. Isack, and Y. Boykov, "Fast approximate energy minimization with label costs," *International Journal of Computer Vision*, vol. 96, no. 1, pp. 1–27, 2012.

[64] S. Wang and J. M. Siskind, "Image segmentation with ratio cut," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 6, pp. 675–690, 2003.

[65] T. Lindeberg, "Scale-space theory: a basic tool for analysing structures at different scales," *Journal of Applied Statistics*, vol. 21, no. 2, pp. 225–270, 1994.

[66] R. Bajcsy, F. Solina, and A. Gupta, "Segmentation versus object representation - are they separable?" University of Pennsylvania, Tech. Rep., 1988.